Simulation of Leaping, Tumbling, Landing, and Balancing Humans

A THESIS

Presented to

The Academic Faculty

By

Wayne L. Wooten

In Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy in Computer Science

Georgia Institute of Technology

March 1998

Simulation of Leaping, Tumbling, Landing, and Balancing Humans

Approved:

Jessica Hodgins, Chairman

Chris Atkeson

Michael Cohen

Norberto Ezquerra

Michiel van de Panne

Date approved by Chairman _____

Acknowledgments

I would like to thank all of the members of the Animation Lab for their support and encouragement. I would especially like to thank Jessica Hodgins for providing a fun and wonderful place to work and for her infinite patience and insistence on the highest possible quality. I would like to thank Dave Brogan and Nancy Pollard for the images they provided in this thesis. I would also like to thank James O'Brien, Ron Metoyer, Victor Zordan and Debbie Carlson for helping develop the software systems described in this thesis. Thanks to Jack Freeman for helping produce the SIGGRAPH animations.

The folks at the Graphics, Visualization & Usability Center provided an exciting environment in which to do my research. I would like to thank Jim Foley for his wise advice and intelligent suggestions. Joan Morton deserves thanks for helping erase administrative problems. I would like to thank the CNS department, especially Randy Carpenter who always quickly addressed systems problems whenever they arose.

Thanks goes to John Martinez for his encouragement and for proofreading the final draft. Generous thanks to Mark Lively at the Bowman Gray School of Medicine for providing me an office and network connection to write the majority of the text. Many thanks to my parents John and Shirley, especially my father who taught me the wonders of science and math from an early age. Finally I would like to thank my wife, Karen, for all the support, encouragement, and love she has provided over the years.

Contents

List of Figures

List of Tables

Summary

This thesis describes an approach for generating transitions between simulated human behaviors in which the designer concentrates effort on the creation of parameterized basis behaviors that can be combined together in an automatic fashion. The parameterization allows the generation of a wide variety of motions from a single basis behavior. If the behaviors are well designed, the exit states of one leaves the simulated character in a valid initial state for the next. This nesting of the input and output states allows easy transitions between behaviors and the generation of many complex behaviors from a small set of basis behaviors. I demonstrate this approach with four basis behaviors: leaping, tumbling, landing, and balancing. Each parameterized control system allows the user to specify properties of the desired behavior such as how high or far to jump and the number of somersaults to perform. I demonstrate transitions between the basis controllers by generating a diverse set of behaviors, including a standing broad jump, vertical leap, forward somersault, backward somersault, back handspring, and various platform dives.

CHAPTER 1

Introduction

Simulations of human motion would be useful for creating realistic, animated characters for movies and commercials, developing video games with autonomous characters that respond appropriately to unpredictable user input, and giving coaches insight into their sport through visualizations of simulated athletes. However, before an adequate source of motion can be produced for any of these applications, methods must be developed that make it easier to generate new behaviors, adapt existing behaviors to new body types, and create transitions between behaviors. This thesis focuses on the last of these problems: transitions between behaviors.

Animators generally use one of three techniques to generate realistic and natural-looking human motion for highly dynamic behaviors: keyframing, motion capture, or dynamic simulation. When keyframing is used, the animator specifies the location of each body segment at key frames in the animation sequence. In traditional cel animation, artists draw the frames in between key frames but with computer-assisted keyframe animation, interpolation methods are used to calculate the body segment positions for the intervening frames. Inverse kinematics and other procedural methods simplify the keyframing process by providing constraints on the motion of hierarchical linkages. With motion capture, the computer determines the three-dimensional location of reflective markers placed on the joints of a live actor. The motion of the joints of a synthetic actor are derived from the locations of the markers as they move over time. With simulation, the equations of motion for a rigid body model of the synthetic actor are combined with control systems to generate motion trajectories. The

control systems take input from high-level specifications and allow the synthetic actors to move autonomously in an environment by avoiding obstacles, changing speed, and interacting with other actors.

Simulation may prove to have advantages over motion capture and keyframing. Using simulation, the lower-level details of the motion are automatically generated by the equations of motion and the control systems, whereas in keyframing the animator must specify many of the details by hand. Simulation generates physically realistic motion while keyframing relies on the skill of the animator to specify realistic or appealing motion. Animation based on motion capture data appears realistic because it is recorded from a live actor moving in the physical world, but the capture process may introduce flaws because of the limited resolution and accuracy of the sensors. Simulation also enables creation of autonomous actors that can react to users or the environment using higher-level control systems to provide the basic behaviors such as running, walking, leaping, or standing. Each basic behavior is more general than an equivalent set of motion capture data or keyframed actions because parameterized behaviors can produce a range of different motions. For example, a control system could produce natural-looking leaping motion for a range of heights from 0.1 m to 1.0 m, but motion capture or keyframing would require many data sets or sophisticated interpolation methods to cover the same range of heights.

Despite the potential advantages of simulation, the difficulty of designing control systems has prevented them from being used extensively. Robust control systems that generate natural-looking motion are difficult to design by hand and automatic methods are not yet sophisticated enough to generate control systems for most human-like behaviors. When control systems are designed by hand, the user must iteratively adjust the control parameters and control laws until the simulation produces the appropriate trajectory and the simulated character performs the desired task.

Figure 1.1. Images from gymnastic and diving simulations. The gymnast on the left is performing a standing backward somersault. The diver on the right is in the flight phase of a backward $1\frac{1}{2}$ somersault with $2\frac{1}{2}$ twists.

A user might have an easier time producing the desired complex behavior by using well designed, parameterized control systems (designed by hand or automatically), as suggested by van de Panne [van96]. Transitions between a large library of these parameterized control systems can create complex behaviors that might otherwise be difficult to produce using physically based methods. However, generating transitions between behaviors is not easy because, in general, two behaviors can not be combined without the addition of a new behavior that takes the output configuration of the first behavior to the input configuration of the second. I address this problem by developing dynamic simulations that use sets of parameterized basis behaviors such that the exit conditions of each behavior are within the input region of the next behavior. This approach, which can be visualized as a set of nested funnels in state space, makes it easy to generate transitions between the behaviors. Because the basis behaviors are parameterized, each can take on a wide variety of forms, yielding many different complex behaviors or paths through the set of nested funnels in state space. Two of these complex behaviors are illustrated in figure 1.1.

Figure 1.2 illustrates three complementary approaches to generating transitions. The leftmost illustration shows n^2 transitions that fully connect a set of n behaviors.

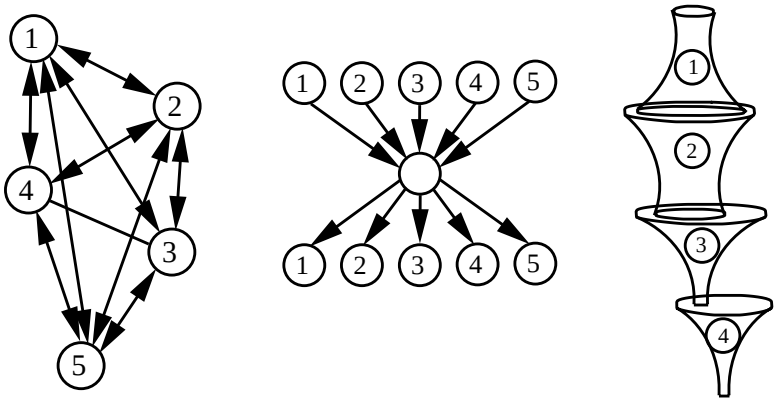


Figure 1.2. Three approaches to generating transitions. On the left is a set of transitions that fully connect the behaviors. The middle illustration shows the use of a single behavior as an intermediate stage in every transition. The final illustration represents each behavior as a funnel in state space. If the exit region of one behavior fits within the capture region of the next, then transitions can be automatically generated between the two behaviors.

Unless the construction of transitions can be automated, it will become more time consuming than developing the behaviors themselves as the set of behaviors grows. A more tractable approach is show in the middle illustration. This approach is often used by video games that use keyframing or motion capture as a source of motion. With this approach, a small set of positions or behaviors are designated as "home positions" and serve as an intermediate stage in each transition. If n behaviors have just one home position then $2n$ transitions are required (from each behavior to the home position and back again). For example, a character in a fighting video game might return to a crouch after each kick or punch.

A third approach, which is explored in this thesis, is to invest the development time in robust, parameterized behaviors rather than in transitions. By increasing the size of the input or capture region for each behavior and the volume of state space occupied by each behavior, one can create a variety of complex behaviors or paths through state space. In the third illustration of figure 1.2, I have represented

each behavior as a funnel in state space. For a given set of parameter values, the behavior moves the character from the input state to a particular point in the exit region. If the exit region of one behavior is contained in the capture region for the next behavior, then generating a transition between these two behaviors is automatic. If the overall size of the funnel is large, then the behavior itself encompasses a wide variety of motion. This approach to generating transitions was inspired by the work of Burridge, Rizzi, and Koditschek [BRK95].

1.1 Problem Statement

I address the problem of generating transitions between behaviors by developing a set of basis controllers for four discrete behaviors: leaping, tumbling, landing, and balancing. Each basis control system provides a set of relatively high-level parameters for generating a variety of different forms of the behavior. Physically realistic motion will be guaranteed within the accuracy of the dynamic model. I attempt to make the resulting motion look natural by using principles from biological systems for inspiration in the design of the basis control systems. I chose this particular set of basis controllers because numerous athletic and gymnastic maneuvers can be constructed by transitioning between these four behaviors.

The leaping controller propels the character into the air. The user or animator selects the height and the horizontal distance of the leap. The control system adjusts the height of the leap by changing the initial height of the hips and adjusts the horizontal distance by changing the location of the projection of the center of mass on the ground with respect to the area of support. Modifying the orientation and magnitude of the ground reaction force alters the character's angular velocity at lift-off.

The tumbling controller produces aerial maneuvers from combinations of som-

ersaults and twists in pike, tuck, or layout positions. Although angular momentum is conserved during flight, the tumbling controller can modify the angular velocities by adjusting the tightness of the tuck or by transferring angular velocity from one axis to another.

The landing controller takes the character from the aerial phase to a state suitable for the balance controller by removing most of the kinetic energy. Forward and rotational velocities are reduced by positioning the legs before touchdown in such a way that the velocities of the center of mass will be near zero when the projected center of mass approaches the center of the area of support. Vertical velocity is reduced by bending the knees and hips after contact.

The balance controller prevents the character from falling down by keeping the projection of the character's center of mass within the area of support. By specifying the height of the hips above the ground and the pitch angle of the upper body, the animator can cause the character to bend over or crouch down while the controller automatically maintains balance.

I demonstrate the utility of transitions between basis controllers by generating a diverse set of behaviors for each character, including four 10 meter platform dives, standing forward and backward somersaults, a handspring, and various leaping maneuvers. I evaluate the behaviors by comparing desired and actual performance and by comparing the animated behaviors with video footage of people performing the same tasks.

1.2 Contributions

The ultimate goal of this research is to produce an autonomous human character that moves in a natural, physically realistic fashion in response to unpredictable user input. The work presented in this thesis has made the following contributions:

- The development of a 56 degree of freedom dynamic simulation of a male and female character and a model that reasonably approximates the appearance and dynamic properties of both characters.

- Four parameterized basis control systems that allow the dynamic simulations to leap, tumble, land, and balance. Each of the basis controllers' parameters can be modified to produce a range of behaviors.

- Basis controllers that allow easy transitions. The final position in state space of one control system normally falls within the range of acceptable initial conditions for the next control system.

- The creation of new, dynamically simulated behaviors including vertical leaps, broad jumps, a forward somersault, a backward somersault, a back handspring, and various platform dives.

- The development of a method that can be further extended by creating additional basis controllers. The new controllers can be combined with the four I have developed to produce a wider variety of complex behaviors.

1.3 Overview of the Thesis

Chapter 2 provides a summary of relevant research in the fields of computer graphics, robotics, and biomechanics. Chapter 3 provides a description of the simulation system and a general description of control systems. Chapters 4, 5, 6, and 7 provide a detailed description and evaluation of the leaping, tumbling, landing, and balancing basis controllers. Chapter 8 describes transitions between basis controllers and provides examples of complex behaviors created with combinations of basis controllers. Chapter 9 summarizes the thesis, and presents possibilities for future research.

CHAPTER 2

Background

The motion of articulated figures has been a subject of study in computer graphics, robotics, and biomechanics for many years. Computer graphics researchers are interested in methods that can be used to simplify the animation of articulated figures. Robotics researchers have developed control systems that enable both physical robots and dynamic simulations to perform complex motion tasks. Biomechanists are interested in the study of human motion and have collected large amounts of kinematic and dynamic data. In the next three sections, I will discuss relevant background material from each of these fields.

2.1 Computer Graphics

Researchers in the field of computer graphics and animation have explored physically based simulation as a method for generating articulated character motion. One of the first systems for simulating articulated figures in the computer graphics literature was developed by Armstrong, Green, and Lake [AGL87]. Their system addressed the problem of computational complexity by using a formulation for the equations of motion that grew linearly with the number of links in the articulated figure. Wilhelms developed a similar system to reduce the complexity of the computation by using the Gibbs-Appell formulation [Wil87]. Neither system gave the user intuitive control over the figure's motion because they required user specification of joint torques. Joint torques are not an intuitive language for the specification of high-level, coordinated

motion. Isaacs and Cohen presented a control methodology that used kinematic constraints and inverse dynamics to determine the forces required to control articulated figures [IC87]. However, this method requires that body segment trajectories, or close approximations, be known in advance. At the time when Isaacs and Cohen developed this approach, obtaining body segment trajectories in advance was not practical, however with the introduction of motion capture systems, this approach is now becoming feasible.

During the late 1980s, computing resources were inadequate for interactive simulations of complex articulated figures. To work around this difficulty, Girard developed a system called PODA that consumed less computational power [Gir87]. Figure 2.1A shows a graphical representation of an articulated figure animated with PODA. Girard's system used inverse kinematics to position the body segments based on the desired location of the end effector. The animator would keyframe the motion of the end effector in a low-level fashion, but would not have to specify the motion of each body segment. Girard also used dynamics to compute the motion of the center of mass of the figure, allowing it to accelerate under gravity while in flight. Girard has recently extended this work and used it to develop a commercial product called "Biped" for the 3D Studio animation system.

Another early animation system, *Jack*, was developed at the University of Pennsylvania by Badler and his colleagues [BPW93]. *Jack* formulated the inverse kinematics of the human figure as a nonlinear programming problem and positioned the body segments based on the location of end effectors. Control programs ensured that the appearance of balance was maintained by moving the feet when the projection of the center of mass onto the ground plane moved outside the polygon defined by the points of contact of both feet. Many enhancements have been added to *Jack* over the years, including a strength model to constrain the motion based on muscle strength

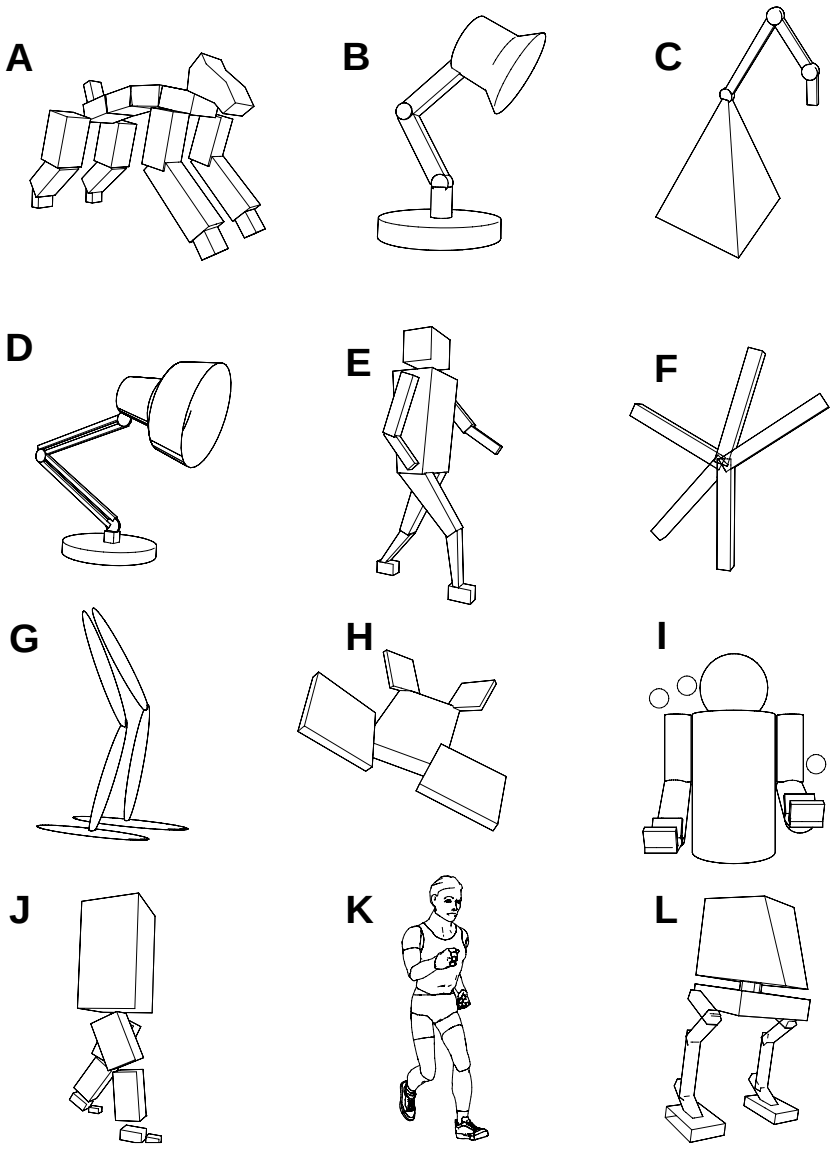


Figure 2.1. A diagram showcasing creatures presented in computer graphics research. **A**: Girard [Gir87]; **B**: Witkin and Kass [WK88]; **C**: Cohen [Coh92]; **D**: van de Panne et al. [vFV90]; **E**: van de Panne et al. [vFV92]; **F**: Ngo and Marks [NM93]; **G**: van de Panne and Fiume [vF93]; **H**: Sims [Sim94b]; **I**: Hodgins et al. [HSL92]; **J**: Stewart and Cremer [SC92] **K**: Hodgins et al. [HWBO95] **L**: Laszlo et al. [LvF96]

and improve the natural look of the motion [LWZB90]. Ko and Badler developed a generalized model of walking using motion captured walking data and inverse dynamics [KB93]. With their generalization, motion could be generated for models of arbitrary anthropometry by using step lengths other than those obtained from motion capture data.

2.1.1 Trajectory Optimization and Optimal Control

After dynamic simulation algorithms were introduced to the computer graphics community, researchers began to turn their attention towards control. Witkin and Kass presented one solution to the problem of automatically generating natural-looking and physically realistic motion for articulated figures. Their method, which they called spacetime constraints, determined a trajectory for each degree of freedom in the system using a two point boundary formulation [WK88]. Constraints were used to specify a desired behavior for the system and to enforce the Newtonian laws of physics (thereby ensuring physical realism). The constrained optimization problem was solved numerically and the result was an optimal, natural, and physically realistic trajectory curve for each degree of freedom. Witkin and Kass used this approach to animate an articulated lamp with six degrees of freedom (figure 2.1B). The lamp's optimization criteria minimized the power consumed by the joint actuators. The result was natural-looking motion for basic, hurdle, and ski jumps. The lamp exhibited a behavior that has traditionally been called anticipation because it crouched down before leaping.

In 1992, Cohen refined the spacetime constraints technique by allowing users to interactively specify constraints and objective functions [Coh92]. The system subdivided spacetime into windows and allowed optimization to take place over subsets of spacetime selected by the user. Cubic B-splines were used to define a C^2 con-

tinuous function describing the trajectory of each degree of freedom. In 1994, Liu, Gortler, and Cohen improved the speed of the spacetime algorithm using hierarchical wavelets [LGC94]. The nature of the wavelet formulation allowed trajectories to have detail only where needed. The wavelet formulation led to better conditioned systems, which in turn allowed fast convergence. They also used common subexpression elimination to reduce the number of expressions used to describe the system, constraints, and objective functions. The creature used to demonstrate the wavelet formulation had six degrees of freedom and is shown in figure 2.1C.

Brotman and Netravali presented an alternative to spacetime constraints [BN88]. They used a linear-quadratic (LQ) optimal control system to apply the minimal amount of control energy needed to ensure that rigid bodies passed through user-specified keypoints. This technique is different from the spacetime approach because it calculates a control value that affects the motion instead of computing the trajectories themselves. The examples they presented were non-articulated, single-body systems—a truck moving along a straight line and an aircraft moving along its equilibrium flight direction. For the truck, the control value corresponded to acceleration or deceleration. For the aircraft, the control value altered the orientation of the plane along the flight path. Even though Brotman and Netravali did not present examples of articulated figures or nonlinear systems, they did mention that their methods could be used in more complex applications by linearizing the problem first.

At the University of Toronto, van de Panne, Fiume, and Vranesic developed a method similar to Brotman and Netravali's [vFV90]. Instead of using an LQ optimal control system to produce the motion of the simulation, they derived a controller using dynamic programming and an optimization criteria of minimum energy consumption. The derived controller could then be reused to guide the simulation from any initial state to the final desired goal state. They used this method to generate the motion for

a lamp with five degrees of freedom (figure 2.1D). Unfortunately the cost of dynamic programming solutions are exponential in the number of degrees of freedom and current algorithms become prohibitively expensive in both time and space as the size of the state space becomes large.

Pai developed a model of a human-like walker based on the notion of least constraint [Pai90]. Least constraint generates motion using inequality constraints, which are time and state based assertions about the allowed states of the simulation. The set of constraints are satisfied using a nonlinear extension of the relaxation method. Pai used a dynamic simulation system called *Newton* [CS89] and a set of constraints to generate motion for a walker with 28 degrees of freedom. The control program included constraints that ensured the foot cleared the ground during the swing phase, the pelvis remained above a certain height, and the placement of the swing foot was suitable for dynamic balance.

In 1992, van de Panne, Fiume, and Vranesic developed a simulation of a two-dimensional, human-like biped that walked up and down gently sloping terrain and stairs [vFV92]. The controller for the stance leg was generated with a state-space controller using dynamic programming [vFV90]. All of the other body segments were controlled with proportional-derivative controllers. The control tables generated by the state-space controller allowed the simulation to ascend and descend stairs and slopes and walk over level terrain. Figure 2.1E shows a diagram of the walking creature.

Ngo and Marks used dynamics to simulate planar creatures with three to five links that were controlled by stimulus-response systems [NM93]. An example of a five link creature is shown in figure 2.1F. The control system for each creature was composed of response functions that generated motion based on the input of the stimulus functions. The stimulus functions were used to sense conditions in the

environment, like contact with the ground, and the response functions changed the configuration of the creature over time. The parameters for the stimulus-response system were generated with a trial and error genetic search algorithm. Optimization criteria, such as longest distance traveled or highest jump achieved, were used to select the best stimulus-response controllers.

In 1993, van de Panne and Fiume developed a similar method that used sensor-actuator networks [vF93]. They modeled ten planar creatures composed of two to six links. Their walking creature with eight degrees of freedom is shown in figure 2.1G. Sensors determined ground contact, joint angles, and joint lengths. The joint actuators used proportional-derivative servos to provide motion control and were connected to the sensors through a network of hidden nodes. The networks were similar to artificial neural networks with weights existing between connections of the sensors, hidden nodes, and actuators. Both simulated annealing and stochastic gradient ascent were used to search for optimal weights for the sensor-actuator network. The evaluation metric for the search methods attempted to generate creatures that could travel the farthest or follow a moving point in space.

Sims presented a system that generated not only a control system using genetic algorithms, but the creature's morphology as well [Sim94b, Sim94a]. The creatures were three-dimensional and had many degrees of freedom. Sensors measured joint angles, contact between linkages, and determined the location of light sources. The control programs were based on a dataflow computer paradigm instead of an artificial neural network. Each dataflow node could take up to three inputs and perform arbitrary computations on them to produce an output. The number of nodes and the connections between them were derived using a genetic algorithm. A number of objective functions were used by the genetic search algorithm to select different behaviors. The algorithm produced creatures that could swim, crawl, jump, and

follow light sources. An example of a swimming creature is illustrated in figure 2.1H.

Grzeszczuk and Terzopoulos automatically generated controllers for flexible creatures such as fish, snakes, and sting rays [GT95, TT94]. The creatures were modeled with point masses connected by springs that acted as actuators. Discrete time and frequency domain controllers were found by optimizing an objective function composed of two terms, one for the amount of control energy used and the other for the smoothness of the trajectory. Both simulated annealing and the simplex method were used to search for the optimal controllers. Higher level control was achieved by combining and smoothly blending the automatically derived lower level controllers.

Many problems must be addressed before automatically generated control systems can be used for simulations with many degrees of freedom. When using search techniques, users have little direct control over the desired motion and must be content with motion trajectories derived through a combination of optimization criteria and repeated iterations of search algorithms. Most of the methods mentioned above either produce natural-looking motion for creatures with few degrees of freedom or produce unnatural-looking motion for creatures with many degrees of freedom. Often natural-looking motion is produced when a simple energy minimization function is used as the optimization criteria but this will not produce many of the movements seen in more complex systems such as human figures. For example, energy minimization for a highly dynamic behavior like a handspring vault might produce natural-looking motion (there are a limited number of ways to perform a vault) but energy minimization for a mime would probably not produce a natural-looking result because style is much more important.

2.1.2 Hand-Designed Control Systems

An alternative to automatically generated control systems are hand-designed control systems. Bruderlin and Calvert developed a simulation of a walking human using a hand-designed control system for a simple leg and biological data for parameters such as pelvis rotation [BC89]. The stance leg was modeled as an inverted double pendulum, where the upper segment represented the torso. The swing leg was modeled as a double pendulum, where the upper and lower leg were modeled by separate segments. The system used a set of control algorithms to determine the forces and torques required to produce walking motion for the simple leg model, and then superimposed a kinematic leg over the simple one. The upper body motion was added kinematically after the motion of the legs had been computed. The system allowed the animator to adjust parameters of the walking simulation such as forward velocity, step length, step frequency, and pelvic tilt.

McKenna and Zeltzer developed a simulation of a six-legged, articulated insect with a gait controller that coordinated the movement of the six legs [MZ90]. The gait controller, derived from observations of gait patterns of walking insects, issued commands to lower level motor controllers. The motor controllers generated forces using exponential springs and were responsible for maintaining support and generating forward motion. Using a framework similar to the one used for the insect simulations, McKenna and Zeltzer also designed a 90 degree-of-freedom human figure model [MZ96]. They used inverse dynamics and feedforward control to compute the forces and torques needed to produce a standing stable posture and the swing phase of a single leg during a walk.

Raibert and Hodgins developed dynamic simulations of a biped robot, a quadruped robot, and a planar kangaroo [RH91]. Control systems composed of state machines and hand-tuned, proportional-derivative servos were used to generate motion for run-

ning, trotting, bounding, galloping, and jumping. The user could control the speed, gait, and path of the simulated creatures, and the motion was used to produce the animation, "On the Run," shown in the 1991 SIGGRAPH Electronic Theater. Hodgins, Sweeney, and Lawrence built on this work when they developed feedback control techniques for several behaviors of human-like creatures: swinging, juggling balls and clubs, bouncing on a see-saw, and riding a unicycle [HSL92]. A diagram of the juggler is shown in figure 2.1I.

Stewart and Cremer developed a simulation system based on *Newton* and they used control systems similar in basic design to those of Raibert and Hodgins [SC92]. The animator used state machines and proportional-derivative servos to control the motion of the simulation. Stewart and Cremer used the system to simulate a biped with fourteen degrees of freedom that could walk in a straight line and up and down stairs. A diagram of the biped model can be seen in figure 2.1J.

Recently, Hodgins and colleagues developed simulations of human athletes using a three-dimensional dynamic model of the human body [Hod94, WH95, HWBO95]. Diving, vaulting, cycling, and running were simulated using a collection of control techniques that included state machines to activate the control laws required for a specific phase of each behavior, synergies to control several degrees of freedom simultaneously, inverse kinematics to derive desired joint angles given desired end effector positions, and proportional-derivative servos to determine the torques required to move joints to the desired positions. A figure of the runner is shown in figure 2.1K.

Laszlo, van de Panne, and Fiume recently presented a technique to add closed-loop feedback to open-loop periodic motions [LvF96]. They used state machines and proportional-derivative servos to develop a walking biped simulation with 19 degrees of freedom. A basic, open-loop walk cycle was achieved using proportional-derivative control to attain a desired joint configuration at each state. Limit cycle control was

then used to modify hip pitch and roll to provide closed-loop stability for the walking motion. Whenever user parameters in the open-loop system were modified, limit cycle control would automatically provide closed-loop stability. Additional animation parameters were used to control the speed, direction, and style of the open-loop walking controller. A figure of the walking biped can be seen in figure 2.1L.

Hand-designed control systems have been used to produce motion for creatures with many degrees of freedom, but natural-looking motion requires careful design of the control system and a good understanding of the desired behavior. All of the control systems discussed above performed single tasks. Much time and effort will be required to develop control systems that perform a wide variety of tasks in a robust fashion.

2.2 Robotics and Control

Many of the ideas used for the control of articulated figures have been borrowed from control techniques developed for robots. Raibert and his colleagues built a series of two-dimensional and three-dimensional running machines that performed a number of dynamic tasks [Rai86]. The control algorithms allowed two-legged and four-legged robots to run, hop in place, change gaits, and run up and down stairs [RCJ82,Hod91, HR91]. Many of the control algorithms developed for the biped robots are relevant to dynamic simulations of human figures. For instance, the notion of adjusting the touchdown position of the foot to control forward velocity has been refined for use in controlling the running speed of a human model [HWBO95].

Hodgins and Raibert developed control algorithms that allowed a planar biped robot to perform a forward somersault [HR90]. The robot's control system performed this maneuver with a series of control states: approach, hurdle, flip, landing, and recovery. Hodgins and Raibert related the control strategies for the flip to possible

biological control strategies such as open-loop control and the motor-tape model. The biped's pitch rate was not controlled during flight, and the success of the maneuver was dependent on control actions taken during ground contact preceding the flight phase. Playter and Raibert developed control algorithms that allowed a three-dimensional biped robot to perform a forward somersault [PR92a, PR92b]. They implemented a tuck servo to alter the rate of rotation in flight to produce better landing conditions. The tuck servo was not compared to open-loop control to determine whether the tuck servo reliably produced better landings.

Robotics researchers have also explored optimal control systems for simulations of planar platform diving. Murthy and Keerthi presented an optimal control system for a two-dimensional diver with four degrees of freedom [MK93]. They formulated a time-optimal control problem using state space and control constraints with a numerical approach to compute the solution. The simulated diver performed both forward and backward somersaults. Solutions required about 10 minutes of computation on an Intel 486-based machine. Crawford and Sastry developed an adaptive controller to search a restricted control space for a planar diver with five degrees of freedom [CS95]. The adaptive control system searched the state space for control parameters that resulted in a $1\frac{1}{2}$ somersault pike with a vertical entry. Gradient descent was used to perform the search and random variations in the control parameters were used to prevent local minima from interfering with the search. The results were qualitatively comparable to humans performing a $1\frac{1}{2}$ somersault pike.

The concept of combining collections of controllers to produce complex behaviors was inspired by the research of Burridge, Rizzi, and Koditschek [BRK95]. Burridge and his colleagues have developed a series of controllers that enable a three degree-of-freedom robot to juggle a ball in a space occupied by obstacles. They developed multiple controllers, each with different domains of attraction, to take the system

from an initial state to a goal state. Then using the notion of preimage back-chaining, they combined the controllers in such a way that the combined (local) domains of attraction covered a large portion of state space of the system. A switching scheme was used to activate the the appropriate controller which moved the system towards a final, global goal state.

2.3 Biomechanics

Biomechanists are interested in the principles of mechanics as applied to biological systems. Kinesiologists are specifically interested in determining how interactions between a biological system and the environment affect the motion of the biological system. Using force plates, video and film digitizers, and computer simulations, scientists in these fields collect data and develop models and theories to explain various aspects of biological motion. This section presents biomechanics research related to the design of control systems for leaping, tumbling, landing, and balancing.

2.3.1 Leaping

Maximum-height jumping is an important measure of athletic fitness and has led to the development of many theories and models. Data collected from experiments and mathematical models provides useful insight into how leaping controllers that generate natural-looking motion might be developed.

Pandy and his colleagues developed a model and control system to study maximum-height human jumping [PZSL90]. The model was two dimensional and consisted of four rigid segments connected with revolute joints. Eight musculotendon actuators, modeled using a variation of the Hill muscle model [GH23], were used to apply forces to the skeletal elements. They solved the optimal control problem by assuming the

neural control signals provided either no excitation or full excitation. The height and joint trajectories produced by the simulation were qualitatively similar to those of human athletes performing a maximum-height squat jump.

In 1993, van Soest and his colleagues developed a similar model consisting of four links with six musculotendon actuators [vSBv93]. Soest was concerned with the role of the gastrocnemius muscle in generating a maximum-height vertical leap. His hypothesis was that the gastrocnemius muscle provided more power as a biarticular muscle (crossing two joints) than as a monoarticular muscle (crossing only one). He developed a simulation with the gastrocnemius attached in biarticular fashion and in monoarticular fashion. A muscle model and control system similar to Pandy's was used to apply forces to the skeleton. The model jumped 10 mm higher in biarticular mode than in monoarticular mode with a variation of 0.1 mm over many trials, lending credibility to the hypothesis.

2.3.2 Tumbling

Researchers in biomechanics have analyzed techniques used by humans in performing aerial maneuvers. Yeadon and his colleagues recorded three-dimensional motion of aerial maneuvers with high-speed film and digitized the resulting footage [Yea90, YAH90]. Yeadon developed a dynamic model of the human body with eleven segments and eighteen degrees of freedom. His simulation was designed for the analysis of twisting somersaults and used inverse dynamics to compute the forces at the joints. Yeadon's data from film capture for three test subjects compared favorably with his simulation.

For many years, researchers have debated how cats land on their feet and how humans perform free fall aerial maneuvers. Frohlich presents an enlightening discussion of the techniques used by humans for initiating somersaults and twists [Fro79].

Through simulation and informal human experiments, he demonstrated that humans could perform somersaults and twists using torque generated by pushing on a platform. He also showed how they could perform similar maneuvers with no angular momentum from the platform. These strategies were vital for the development of the tumbling control system described in chapter 6.

2.3.3 Landing

Studies of strategies used in landing and the impact forces produced during landing provide useful theories for the development of control systems. Özgüven and Berme presented a model of impact forces experienced by humans landing from heights of 0.45m and dismounts from a horizontal bar onto regulation gymnastic mats [OB88]. A simple model composed of two masses connected by springs and dampers was developed to predict the maximum ground reaction force experienced under various landing conditions. Parameters for the model were obtained from 12 landing trials. Experimental data showed that the maximum reaction force ranged from 8.2 to 11.6 times body weight for the dismount and from 5.0 to 7.0 times body weight for the vertical landing. The predicted reaction forces were close to the measured reaction forces.

McNitt-Gray studied the kinetics of the lower extremities during drop landings from three heights [MG93]. She found that the geometric configuration of the lower limbs were independent of impact velocity before touchdown and that extensor moments in the hip, knee, and ankle increased in response to increased impact velocity. She also found that professional gymnasts tended to dissipate landing load by using larger hip and ankle extensor moments during higher impact velocities, whereas recreational athletes tended to use a greater range of flexion and had longer landing phase durations. Her study suggested the same kinematic strategy was used regard-

less of impact velocity and that only gains were changed to compensate for increased reaction forces.

Caster and Bates assessed the strategies used for landing by four male subjects dropped from a vertical height of 0.6m [CB95]. Their study attempted to determine the relative contributions of passive control (mechanical) and active control (neuromuscular) used during landing. They hypothesized that if weights were added to the subject's ankles and the ground reaction force increased in proportion to the weight added, then only mechanical strategies were being used. However, if the ground reaction force was attenuated, then the subjects used neuromuscular response as a protective measure to avoid large, damaging reaction forces. EMG data was collected to help determine the strategies employed by the subjects. The results showed that both mechanical and neuromuscular strategies were used, depending on the amount of added mass and prior experience.

2.3.4 Balancing

Research into how humans respond to disturbances in balance provides useful data for developing balance control systems. Nashner and his colleagues studied how humans respond to disturbances in balance while standing and walking [Nas83]. They observed that muscles react in response to a disturbance in a distal to proximal order and the shortest latencies in response to a disturbance are 70-100ms. These observations have been used in the design of both the balance and leaping controllers described in chapters 4 and 5. In summarizing Nashner's research, Ghez [Ghe77] emphasized that postural stability in humans is maintained by a combination of preprogrammed feedforward responses and compensatory feedback responses. Nashner observed that after only a few repeated trials of similar disturbances, the preprogrammed response was attenuated [Nas76].

Winter, Patla, and Frank provided a comprehensive review of the balance control literature in addition to presenting findings of their own [WPF88]. One of their interesting observations was that the nervous system anticipates disturbances to balance caused by voluntary movements and applies corrective action to compensate for the disturbance. When subjects' perturbed their balance by pulling on a fixed handle, EMG data showed that the gastrocnemius and hamstrings reacted in a distal to proximal order before the bicep muscle began to pull on the handle.

2.4 Summary

In this chapter, I presented background material from computer graphics, robotics, and biomechanics, three fields that are relevant to the creation of controllers for dynamically simulated human figures. The controllers presented in this thesis draw most heavily from research on hand-designed control systems and control systems for bipedal characters. Each of the basis controllers described in chapters 4-7 are based on the corresponding research presented in the biomechanics section.

CHAPTER 3

Simulation System

This chapter describes the software environment used to produce the animations presented in this thesis. The user begins a simulation session by loading a file defining the initial state of the system. The user specifies the character's behavior by modifying control system parameters via the user interface or a script file containing multiple parameter-value pairs. Then the user runs the simulation and records a set of time-varying data for the parameters of interest. A set of joint trajectories are simultaneously stored for playback through a graphical representation of the character. This data can then be analyzed to evaluate the performance of the control systems and to gain insight into parameter changes that might improve performance. When the user is satisfied with the simulated motion, high quality animation can be produced from the recorded joint trajectories. A diagram illustrating the major components of the environment can be seen in figure 3.1.

The first section of this chapter provides a description of the dynamic simulations. The second section describes the control systems that compute the forces and torques for the dynamic simulations. The final section describes how the simulation data is used to produce animation.

3.1 Dynamic Simulations

The simulation system produces motion for articulated characters using equations of motion that govern the trajectories of connected rigid bodies. A set of parameters

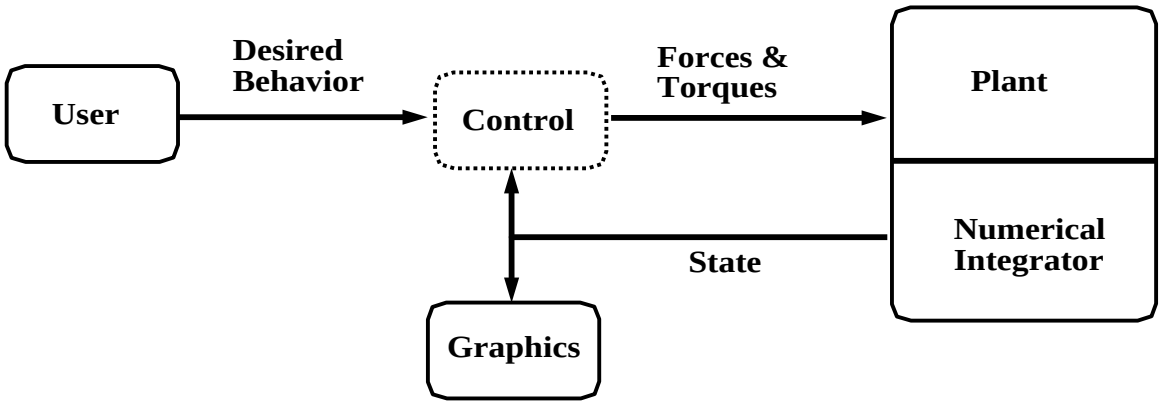


Figure 3.1. A diagram of the animation system used for the dynamic simulation of human characters. The user controls the animation by providing parameters for the control system, such as the amount of twist during lift-off, forward velocity of the torso, or the height of a jump. The control system uses this information to compute the torques that should be applied at each joint. The simulation uses the equations of motion and the current state of the system to compute the joint accelerations for a given set of joint torques. The velocity and position at the next time step are computed by integrating the acceleration. The system state is then used to draw the graphical image and provide feedback for the control system.

for each rigid body segment and each joint is required to derive the equations of motion for a particular character. The parameters for the rigid bodies are mass of the segment, center of mass of the segment, and moment of inertia about the center of mass. Parameters for the joints are location of the joint, number of degrees of freedom at the joint, and bodies connected by the joint.

The equations of motion for the male and female simulations are generated from the model's parameters with a commercially available package called SD/FAST. The package generates C subroutines for the equations of motion using a variant of Kane's method and a symbolic simplification phase [RS86, HRS91]. The subroutines determine the accelerations, velocities, and positions of each body segment given the applied forces and torques. User-defined routines enforce collision constraints to prevent the hands and feet of the character from penetrating the ground plane. A fixed step size, fourth-order Runga-Kutta integrator advances the simulation forward in time.

Convex polyhedral body segments that approximate the shape of each characters' body parts are shown in figure 3.2. The volume and center of mass of each polyhedron is computed using an algorithm developed by Lien and Kajiya for computing integrals over arbitrary non-convex polyhedra [LK84]. Density data for each body segment is combined with the volume and center of mass data to calculate the mass and moments of inertia for the corresponding body segment [DG65, Dem55]. The densities and mass properties of each body segment are provided in tables 3.1 and 3.2. One of the simplifying assumptions of these models is that the density of each body segment is uniform. In general this assumption is incorrect. For example, the upper chest has lower density in the lung tissue than in the muscle tissue. However, the mass and moments of inertia of the body segments compare favorably with the anatomical literature and it is unlikely that this approximation has a noticeable

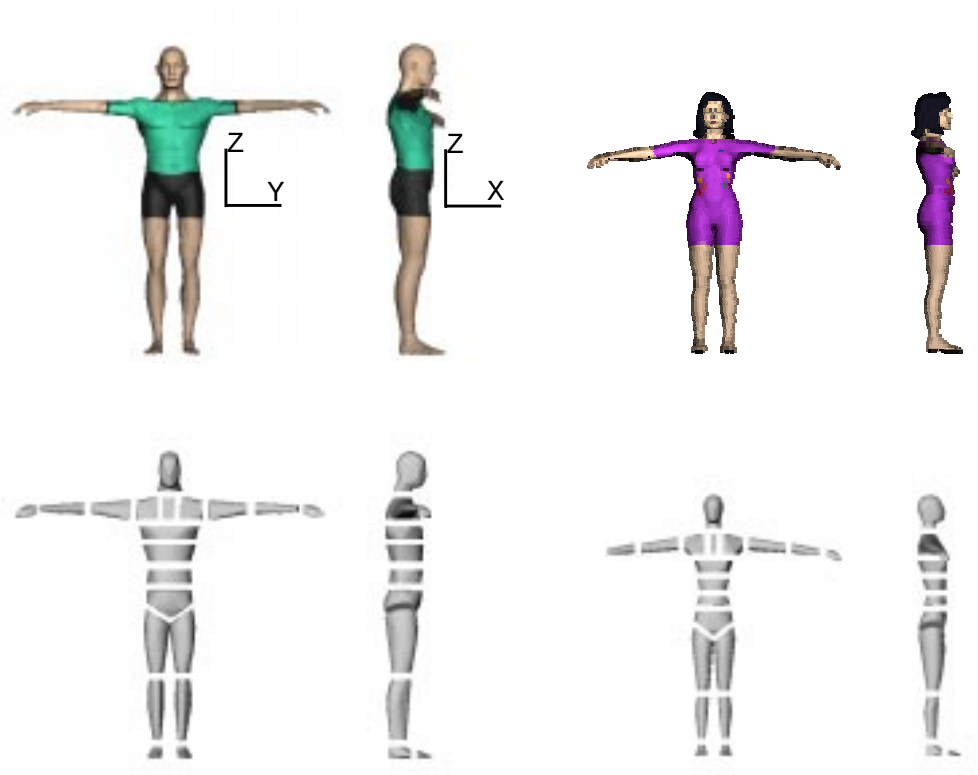


Figure 3.2. The top row of figures show rendered versions of the NURBs models used to develop the dynamic simulations. The bottom row of figures illustrate the relative locations and shapes of individual body segments (the segments have been separated at the joints to illustrate the boundaries of each segment). The segments are closed, three-dimensional convex hulls that approximate the shapes of the equivalent NURBs surfaces. The segments are used to derive the mass properties for the dynamic model.

Link	Density (g/cm^3)	Mass (kg)	Moment of Inertia (x, y, z kgm^2)			Center of Mass (x, y, z m)		
Head	1.171	7.11	0.0492	0.0597	0.0298	0.045	0.000	1.693
Sternum	1.009	2.20	0.0051	0.0123	0.0089	-0.011	0.000	1.469
Upper Torso	1.009	8.78	0.1036	0.0494	0.1401	0.000	0.000	1.353
Mid Torso	1.009	8.39	0.0737	0.0433	0.0956	0.010	0.000	1.245
Lower Torso	1.029	6.16	0.0443	0.0271	0.0608	0.015	0.000	1.129
Pelvis	1.029	10.73	0.0862	0.0711	0.1095	-0.007	0.000	1.002
Upper Leg	1.040	10.46	0.1535	0.1637	0.0443	-0.008	±0.096	0.763
Lower Leg	1.079	5.32	0.0745	0.0784	0.0113	-0.041	±0.096	0.333
Foot	1.066	0.83	0.0008	0.0017	0.0017	-0.020	±0.097	0.039
Metatarsal	1.066	0.28	0.0002	0.0002	0.0003	0.085	±0.103	0.018
Shoulder	1.009	4.09	0.0144	0.0204	0.0204	-0.009	±0.104	1.464
Upper Arm	1.068	3.82	0.0257	0.0092	0.0272	0.005	±0.296	1.467
Lower Arm	1.102	1.78	0.0130	0.0029	0.0148	0.057	±0.591	1.478
Hand	1.070	0.67	0.0012	0.0007	0.0013	0.143	±0.832	1.457

Table 3.1. Dynamics parameters of the rigid body segments of the male model. The total mass of the model is 97.84 kg. The moment of inertia is computed about the center of mass of each link in the configuration shown in figure 3.2. The center of mass is given in world space. The densities are given in Dempster and Gaughran [DG65].

Link	Density (g/cm^3)	Mass (kg)	Moment of Inertia (x, y, z kgm^2)			Center of Mass (x, y, z m)		
Head	1.171	4.63	0.0217	0.0262	0.0143	-0.046	0.000	1.487
Sternum	1.009	1.20	0.0025	0.0054	0.0033	-0.046	0.000	1.270
Upper Chest	1.009	4.28	0.0236	0.0158	0.0337	-0.023	0.000	1.159
Mid Chest	1.009	2.62	0.0103	0.0064	0.0135	-0.014	0.000	1.071
Lower Chest	1.029	2.10	0.0083	0.0043	0.0112	-0.010	0.000	0.991
Pelvis	1.029	6.33	0.0387	0.0283	0.0439	-0.027	0.000	0.888
Upper Leg	1.040	7.41	0.1115	0.1155	0.0220	-0.027	±0.082	0.662
Lower Leg	1.079	2.00	0.0150	0.0152	0.0021	-0.039	±0.087	0.263
Foot	1.066	0.51	0.0004	0.0007	0.0006	-0.015	±0.094	0.036
Metatarsal	1.066	0.15	0.0001	0.0001	0.0001	0.073	±0.087	0.014
Shoulder	1.009	3.07	0.0097	0.0122	0.0121	-0.047	±0.086	1.265
Upper Arm	1.068	1.83	0.0122	0.0022	0.0124	-0.052	±0.296	1.265
Lower Arm	1.102	0.78	0.0027	0.0007	0.0030	-0.005	±0.539	1.235
Hand	1.070	0.33	0.0004	0.0003	0.0004	0.055	±0.718	1.193

Table 3.2. Dynamics parameters of the rigid body segments of the female model. The total mass of the model is 55.24 kg. The moment of inertia is computed about the center of mass of each link in the configuration shown in figure 3.2. The center of mass is given in world space. The densities are the same as the male model. [DG65].

effect on the motion.

The "skeleton" of the character is modeled as a set of rigid links connected by rotary joints with one, two, or three degrees of freedom. Some joints, like the knee, are modeled as a single pin joint whereas others, like the shoulder, are modeled as a ball joint. Tables 3.3 and 3.4 give the coordinates of each joint in model space along with the number of degrees of freedom at that joint. The origin of each model places the heels and balls of the feet at 0.0 meters along the Z-axis, with the character facing the positive X-axis.

3.1.1 External Forces

External forces are applied to points on the hands and feet of the character to prevent them from penetrating the ground plane. Constraint equations are used to compute the reaction forces that keep the feet above the ground and figure 3.3 shows a diagram of the constraint system. Five constraint equations are used for each foot: four keep the four corners of the foot above the ground plane and one prevents the toe from penetrating the ground. The linear acceleration relative to the ground (along the vertical axis) of each contact point is the constraint error. The penetration of the foot into the ground and the velocity of the foot relative to the ground are used to stabilize the foot's constraint error [Bau72]. For the toe, the angular acceleration of the toe relative to the ground is the constraint error. The pitch of the foot relative to the ground and the angular velocity of the toe joint stabilize the toe's constraint error. A force is applied to each contact point on each foot to enforce the constraint that keeps each point above the ground. A torque is applied about the toe joint to keep the toe above the ground. The feet are prevented from slipping by two springs and dampers at each contact point. One prevents the contact point from slipping along the X-axis and the other prevents slipping along the Y-axis. To allow the feet

Joint Name	Joint Location (x, y, z m)			DOFs
Neck (C2)	0.013	0.000	1.555	XYZ
Spine (T4)	-0.012	0.000	1.400	XYZ
Spine (T7)	-0.002	0.000	1.304	XYZ
Spine (L1)	-0.001	0.000	1.180	XYZ
Spine (L3)	0.001	0.000	1.079	XYZ
Hip	-0.019	±0.082	0.937	XYZ
Knee	-0.020	±0.096	0.524	Y
Ankle	-0.054	±0.088	0.084	XY
Metatarsal	0.049	±0.110	0.022	Y
Clavicle	0.003	±0.035	1.483	XZ
Shoulder	-0.001	±0.182	1.483	XYZ
Elbow	0.016	±0.456	1.475	YZ
Wrist	0.119	±0.758	1.474	XY

Table 3.3. The joints and their world space locations for the male model in the configuration shown in figure 3.2. The height of the model is 1.84 meters. The degrees of freedom for each joint are listed with each axis of rotation. Joints having +Y locations are on the left side of the body and those with -Y locations are on the right side of the body.

Joint Name	Joint Location (x, y, z m)			DOFs
Neck (C2)	-0.062	0.000	1.394	XYZ
Spine (T4)	-0.035	0.000	1.201	XYZ
Spine (T7)	-0.031	0.000	1.111	XYZ
Spine (L1)	-0.022	0.000	1.027	XYZ
Spine (L3)	0.016	0.000	0.962	XYZ
Hip	-0.038	±0.077	0.836	XYZ
Knee	-0.027	±0.078	0.401	Y
Ankle	-0.039	±0.088	0.082	XY
Metatarsal	0.043	±0.094	0.015	Y
Clavicle	-0.035	±0.023	1.300	XZ
Shoulder	-0.066	±0.171	1.287	XYZ
Elbow	-0.036	±0.450	1.244	YZ
Wrist	0.038	±0.666	1.220	XY

Table 3.4. The joints and their world space locations for the female model in the configuration shown in figure 3.2. The height of the model is 1.6 meters. The degrees of freedom for each joint are listed with each axis of rotation. Joints having +Y locations are on the left side of the body and those with -Y locations are on the right side of the body.

to leave the ground, the constraint and spring forces are applied only when a point on the foot has penetrated the ground and the vertical velocity of that point relative to the ground is negative.

The foot constraints are adequate for standing and leaping but the hands also contact the ground during handpspring maneuvers. In order to circumvent the design complexity of constraint equations, a penalty-based method is used to ensure that the hands do not penetrate the ground plane. A reaction force is applied to each vertex on the hand model that has penetrated the ground. The magnitude of the reaction force F is proportional to the distance the vertex lies below the ground δ. The direction of the force is opposite the vertexes direction of travel. The reaction force F can be resolved into a normal force F_n and an orthogonal force F_o (figure 3.4).

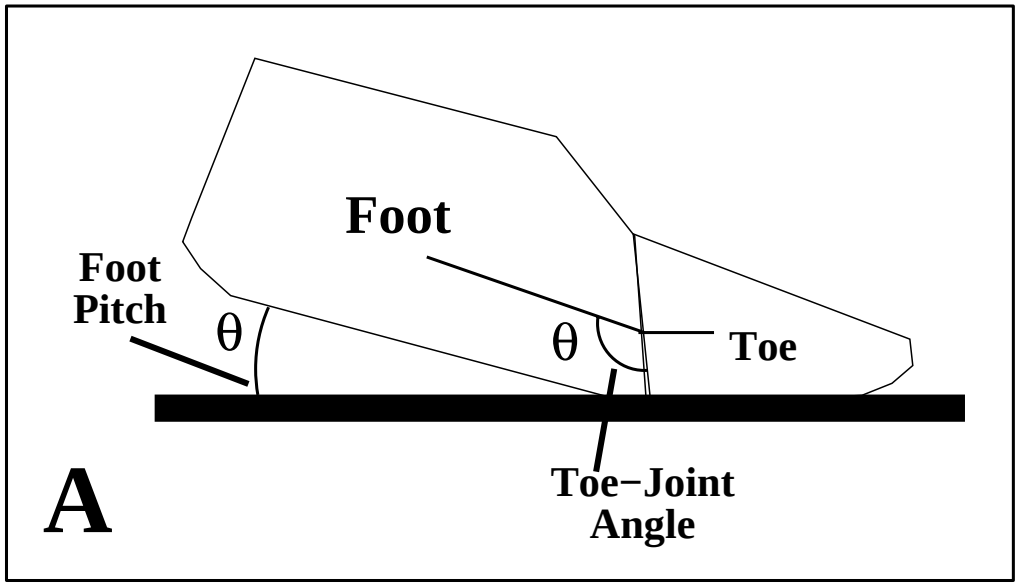
$$F_n = k_p U_n \delta - k_d V_n \tag{3.1}$$

$$F_o = -U_o \mu |F_n| \tag{3.2}$$

where k_p and k_d are penalty position and velocity constants, U_n and U_o are the unit normal and the unit projection of F onto the ground plane, and μ is a sliding friction constant.

3.2 Control Systems

In this simulation environment, multiple control systems, arranged in a three-level hierarchy (figure 3.5), are used to compute forces and torques that move the joints of the character. At the highest level an animator determines the behavior that is to be executed at a specific point in time. The mid-level control systems produce desired joint trajectories to accomplish specific behaviors. Low-level control systems servo each of the character's joints to the desired location using proportional-derivative servos.



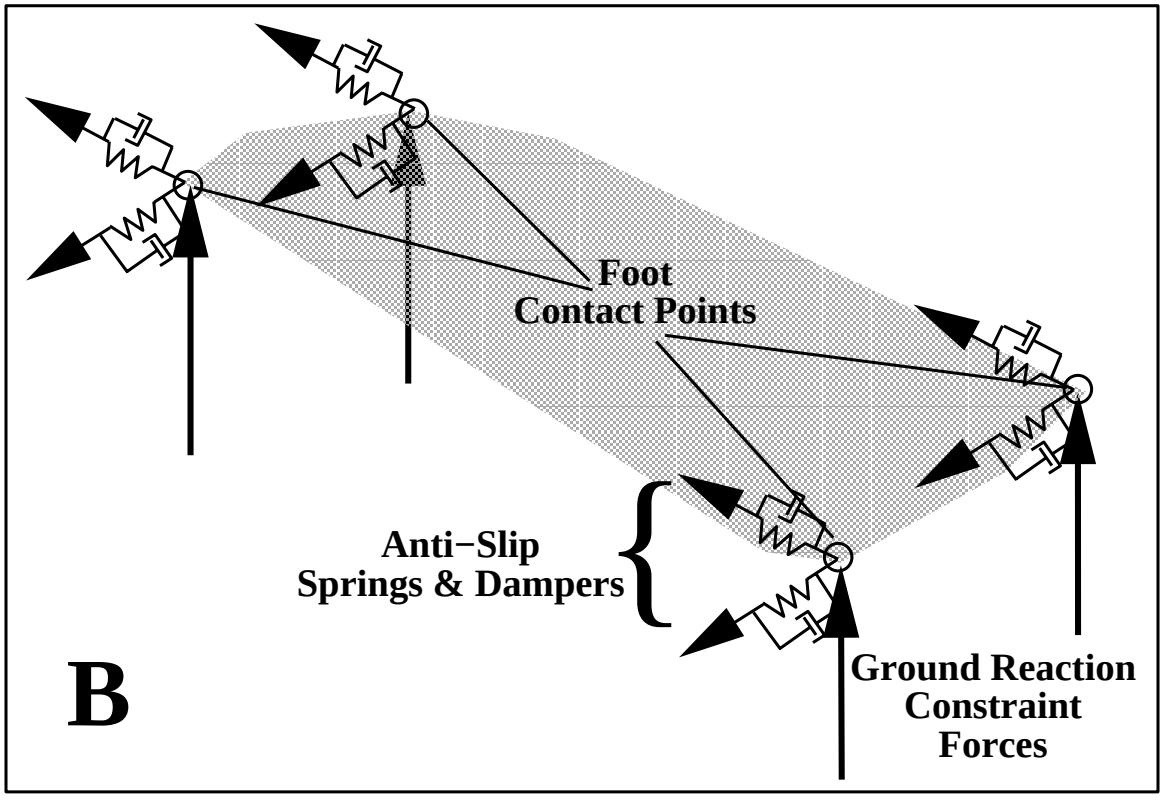


Figure 3.3. **A.** The toe is constrained to remain on the ground by constraining the toe joint angle to be the same as the foot pitch angle. **B.** A combination of constraints, springs, and dampers at each of four contact points ensure that the foot does not penetrate or slip on the ground. The spring and damper constants for the male character are 50000 and 1000, for the female character they are 25000 and 500.

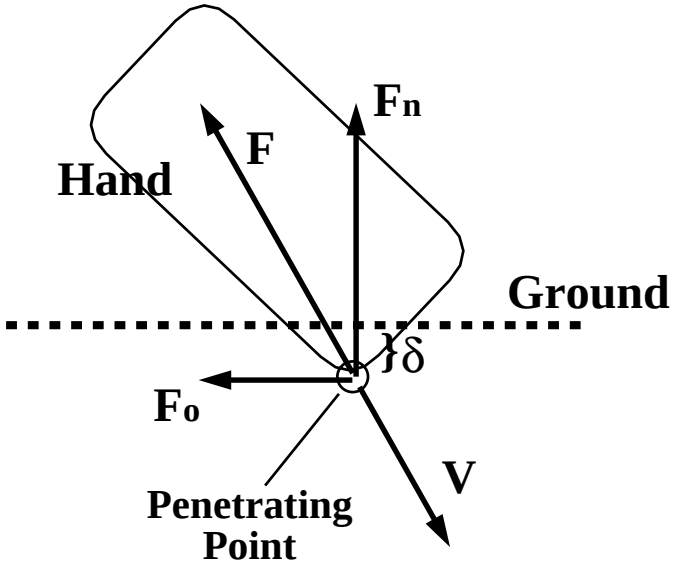


Figure 3.4. To prevent the hand from passing through the ground plane, penalty forces are applied to each point that falls below the ground plane.

The controllers at the highest level sequence individual actions to produce complex behaviors. For example, the high-level controller might concatenate balancing, leaping, and landing actions to produce a standing broad jump. The high-level controller also determines when the system should activate one action and deactivate another. In this simulation environment, scripts are normally written to specify the sequence of actions (along with parameters for each action) that specify the behavior to be performed.

It is the responsibility of the basis controllers to execute and transition between the desired actions specified at the highest level. These mid-level controllers form the core of the work presented in this thesis. They take input parameters and produce desired joint trajectories for the lower-level, proportional-derivative servos. Four mid-level controllers that demonstrate the basis controller concept will be discussed in detail in chapters 4-7.

The low-level control system is used to derive the torques that are applied at each axis of each joint. The lowest level control system positions each degree of

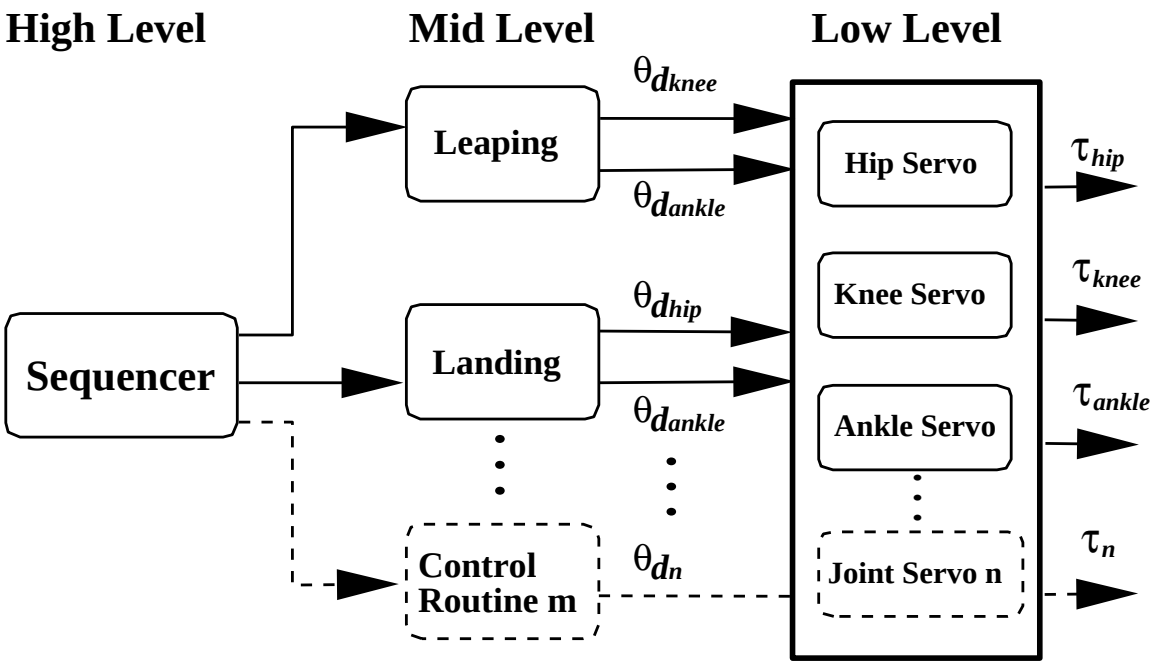


Figure 3.5. This figure illustrates the flow of data between different levels of control. The high-level controller selects a specific mid-level controller to accomplish a given task. Each mid-level controller computes desired joint locations. The desired joint locations are then passed to the set of low-level controllers (proportional-derivative servos for each degree of freedom) which compute the torque required to position each joint.

freedom using a proportional-derivative servo:

$$\tau = k_p(\phi_d - \phi) - k_v\dot{\phi} \tag{3.3}$$

where τ is the joint torque, ϕ is the joint angle, ϕ_d is the desired joint angle, and $\dot{\phi}$ is the joint rotation rate. The gains of the proportional-derivative servo, k_p and k_v, are chosen empirically for each joint. When a user changes the desired angle for a joint, a smooth trajectory is used to take the current desired angle ϕ_d to the new user desired angle ϕ_{u_d}, because eliminating step changes in the desired angles reduces the jerk observed in the generated motion. The smooth trajectory is generated using a Hermite curve, where the time between the current desired angle and the new user desired angle is specified in seconds and the slopes at the two end points are zero.

3.3 Graphical Models

The animator uses the polygonal models shown at the bottom of figure 3.2 to obtain graphical feedback from the simulation system and evaluate the motion from a given set of control parameters. Once the animator is pleased with the motion, the joint trajectories are applied to a more complex geometric representation of the character. This section describes how polygonal and NURBs models are created and used to produce final animation sequences.

The geometric NURBs models used for creating the dynamic simulations of a male and female character were purchased from Viewpoint Datalabs. The female model is composed of 64 NURBs surfaces and the male has 61 surfaces. The models are oriented to face the positive X axis when standing erect.

The NURBs models are used to determine the placement and number of articulated joints for the dynamic simulation and skeleton structure in the Alias-Wavefront animation package. Additional joints can mimic human anatomy more closely, how-

ever the number of joints directly affects the number of degrees of freedom in the dynamic model, the running time of the simulation, and the complexity of the control systems. A trade-off between geometric complexity and computational cost can be made in an attempt to build a model that moves realistically without requiring extremely long simulation times. Instead of placing a three degree-of-freedom joint at every vertebrae in the back (resulting in 72 degrees of freedom), only five are used (resulting in 15 degrees of freedom). Clavicles are added to produce better looking shoulder motion at the cost of eight additional degrees of freedom (two at the sternum and two at the shoulder of each arm). A more precise model would more accurately represent the mechanics of every joint but would produce a much slower simulation.

Once the number of joints and their locations are determined, a simplified polygonal mesh that approximates the surfaces of the NURBs model is created. The polygonal model is created for two reasons: to calculate the dynamic parameters of each body segment and for efficient, interactive rendering. After the surface is polygonalized, individual body segments are created using constructive solid geometry methods (each segment is cut through the center of the joint). In order to achieve accurate results from the integration algorithm mentioned in section 3.1, the polygonal surface must be closed, so a three-dimensional convex hull algorithm is applied to each body segment before performing integration.

Simple, fast, polygonal renderings of the character are produced using kinematic information from the dynamic simulation. The world-space location and orientation of each body segment from the simulation is applied to the appropriate polygonal body segment in an Inventor scene graph. If there is an error in the specification of joint location parameters in the simulation, the developer can visually detect them. However, this technique requires seven parameters to position each segment and the geometry of a fast, polygonal model must be carefully designed to overlap in an

acceptable way while avoiding geometric discontinuities at the joints.

Detailed rendering of the character is produced using Alias-Wavefront. The geometry of the character is attached to an underlying skeleton that is built in a hierarchical fashion. To animate the character, the joint positions of each degree of freedom (obtained from the dynamic simulation) are applied to the corresponding joints of the skeleton. This skeletal representation differs from the world-space representation in that it requires at most three parameters to orient each body segment. The drawback is that errors in the joint location parameters of the dynamic model may not be readily apparent.

The NURBs surfaces representing the skin of the character are attached to the skeleton and deformed as the skeleton moves underneath. The deformations remove the need to overlap joint geometries and produce a detailed model that renders faster than an equivalently detailed polygonal model when using Renderman. Figure 3.6 illustrates the deformation of NURBs surfaces representing the leg as they bend around the knee joint. Each control vertex on the surface of each NURB rotates by an amount proportional to the rotation of the knee joint. Control vertices below the knee receive the same rotation as the knee. Control vertices above the knee receive no rotation. Control vertices around the knee receive a percentage of rotation based on values specified by the animator. The amount of rotation applied to a control vertex may also depend on the axis of rotation; a control vertex that rotates about the hip's X-axis might only receive a 20% rotation, but the same vertex receives a a 60% rotation about the Y-axis. No deformations are performed that model the motion of muscles and fat that lie between the skeleton and skin of the character.

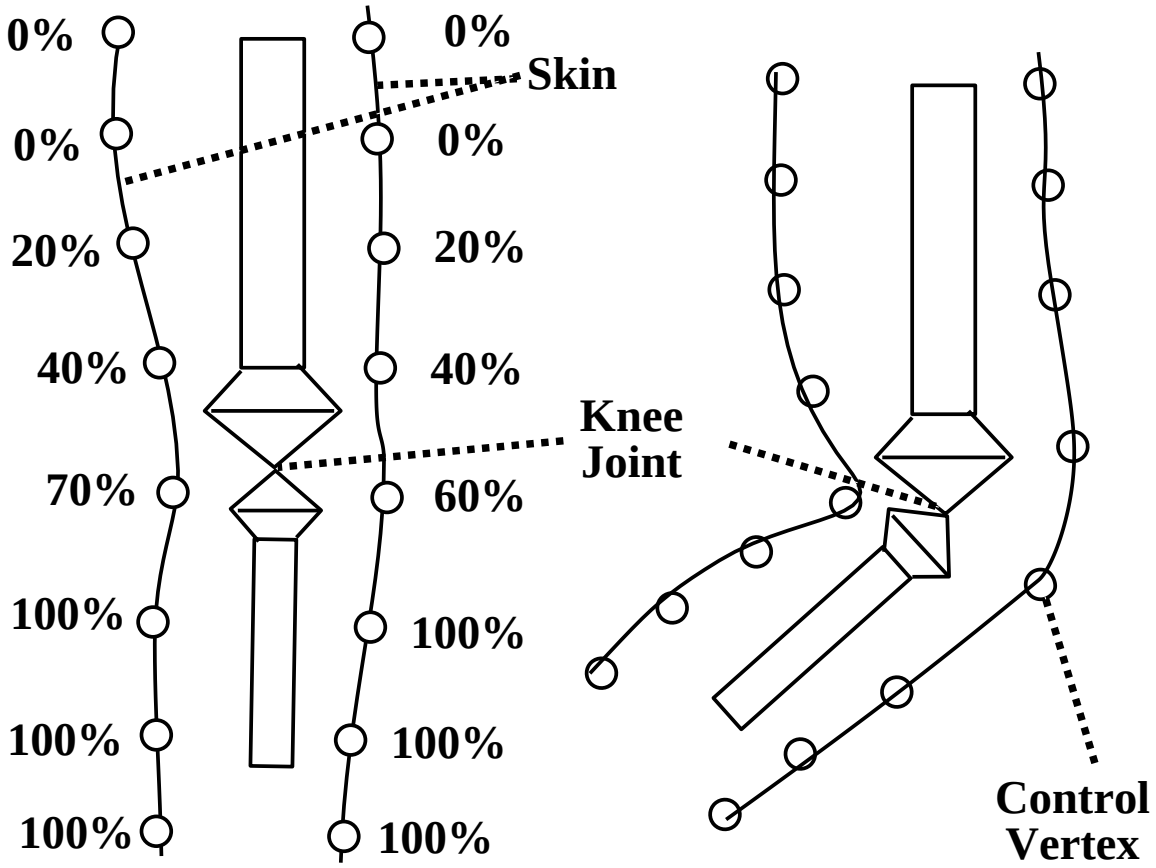


Figure 3.6. This figure illustrates the deformation of the skin of the leg around the knee joint. Control vertices well above the knee are not affected by the rotation; however, those near or below the knee are. The weights affecting the control points do not change as the knee joint bends.

3.4 Summary

In this chapter, I presented the groundwork for the simulation system used to develop the basis controllers in chapters 4-7. The user provides control parameters to the simulation system and the system produces the resulting motion. The equations of motion were created using a commercial package called SD-Fast and the control systems are arranged in a hierarchy of high, mid, and low-level control. Renderings of the human figures were produced using Inventor (for interactive rendering) or Renderman (for detailed rendering). NURBs surfaces were used in final renderings to produce a model that had flexible "skin" about the joints.

CHAPTER 4

Leaping

Chapter 1 described a methodology for combining basis controllers to produce complex behaviors for simulated characters. This chapter describes a basis controller for leaping that is designed to propel simulated characters into the air. This controller allows characters to leap over obstacles, initiate gymnastic maneuvers, or simply leap for joy.

The leaping controller described in this chapter produces a subset of leaps that approximately match human performance. In particular, the control system produces vertical leaps of up to one meter high and standing broad jumps of about two meters. In addition, the controller generates angular momentum which allows the tumbling controller (described in chapter 5) to create different types of somersaults and twists. The first part of this chapter describes and evaluates the controller for leaping vertically, the second section discusses the standing broad jump, and the last section describes methods for altering angular momentum.

4.1 Leaping Vertically

People leap into the air by flexing and then quickly extending their hips, knees, and ankles. The leaping controller mimics this approach to propel the simulated characters into the air. The timing of the joint extension is important to the height of the jump and the controller extends the joints in a proximal to distal order (hip, knee, then ankle) to match the strategy used by humans [PZSL90]. The desired hip angle is set

to reach full extension in 0.1 s, the knee angle in 0.15 s and the ankle angle in 0.22 s.

To initiate a vertical leap, the balance controller (described in chapter 7) first lowers the pelvis of the character to a specified height above the ground (l_{hip}). The controller also bends the torso to a pitch of Δ_{hip} while keeping the location of the center of mass of the body over the area of support with the ankle joints. The leaping controller then increases the spring constants in the ankle joint servo by a factor of 4.45 and begins the extension of the hip, knee, and ankle. The same timing is used for joint extension independent of l_{hip}.

The leaping controller uses the desired leaping height specified by the user to determine appropriate values for the two lower-level control parameters: l_{hip} and Δ_{hip}. A lower l_{hip} increases the range of motion during preparation for the leap, producing more thrust and causing the character to leap higher. A higher leap is also produced by a significant forward pitch in the upper torso (Δ_{hip}). The values for l_{hip} and Δ_{hip} are determined from a set of four to five experimental trials using piecewise linear interpolation (table 4.1). In both models, the use of linear interpolation is adequate to produce acceptable motion in the range from 0.1 m to 1.0 m. Graphs showing the performance of each controller are shown in figure 4.1.

The height of an athlete's vertical leap is used to measure power and physical fitness and is measured as the distance an athlete's fingertips reach above the maximum height reached in the standing position. Male professional basketball players can leap higher than a meter and female volleyball players can leap three quarters of a meter. For example, Reggie Phillips, a member of the Harlem Globetrotters, has a vertical leap of 1.22 m and Joselyn Robbins, who plays volleyball for Hawaii, has a vertical leap of 0.76 m. In comparison, the leaping controller produces leaps of about a meter for both the male and female simulation.

Biomechanists are interested in studying vertical leaps in humans because the

Male

l_{hip}	Δ_{hip}	COM height
0.85	0.0	0.073
0.75	-0.5	0.224
0.65	-1.0	0.476
0.55	-1.5	0.713
0.45	-1.8	0.948

Female

l_{hip}	Δ_{hip}	COM height
0.7	0.0	0.041
0.6	-0.5	0.240
0.5	-1.0	0.648
0.4	-1.5	1.158

Table 4.1. The maximum height of the character's center of mass (COM) during a leap is used to interpolate values for the two control parameters (l_{hip} and Δ_{hip}) when performing a vertical leap.

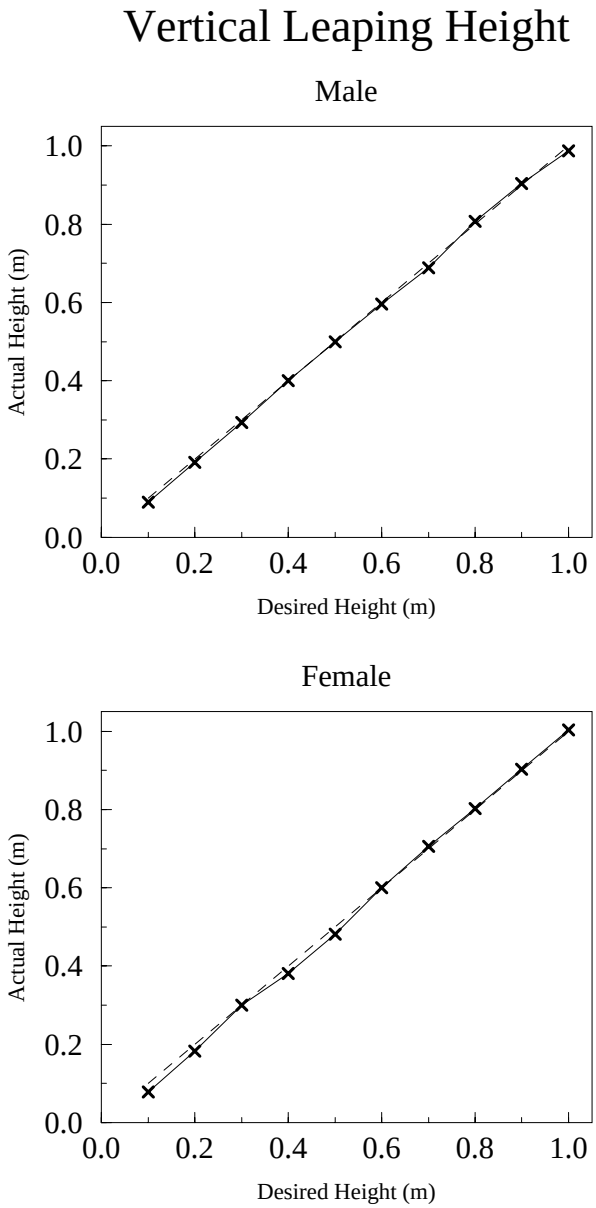


Figure 4.1. Graphs showing the height of the vertical leap achieved by the character in response to a commanded height. The dashed line represents a perfect performance where the achieved height is equal to the desired height.

objective of leaping as high as possible is easily defined and measured. Pandy and his colleagues collected data from a 24-year-old male (weight 754 N) who performed a maximum height vertical jump [PZSL90]. They found that it took about 1.1 s for him to initiate a leap 0.45 m into the air from a standing position. Peak ground reaction forces were about 2.5 times body weight for this height. Ground reaction force graphs for the simulated characters (figure 4.2) show that the male character takes about 1.1 s to leap 0.5 m and that the peak ground reaction force is about 5.8 times body weight. One explanation for the discrepancy in ground reaction force is that the simulation is less efficient at propelling the figure into the air.

4.2 Standing Broad Jump

In the standing broad jump as well as the vertical leap, people first crouch down and and then explosively extend their lower limbs. But unlike the vertical leap, the broad jump also requires substantial forward velocity at lift-off. The timing of the standing broad jump is somewhat slower than that of the vertical leap with the desired values for the hip angles reaching full extension in 0.15 s, the knees in 0.2 s, and the ankles in 0.45 s.

To initiate the broad jump, the balance controller lowers the character's hip to l_{hip} while positioning the character's center of mass forward of the center of the area of support (Δ_{COM}). The leaping controller then increases the spring constant for the ankle servo by a factor of 3.6 and swings the arms forward as the hips, knees, and ankles are extended. The ground reaction force has a significant horizontal component because the character's center of mass is not centered above the area of support (figure 4.3). This forward position of the center of mass also means that when the pelvis reaches the desired height, the center of mass will fall outside of the area of support. The character must then initiate the second phase of the leap (by rapidly

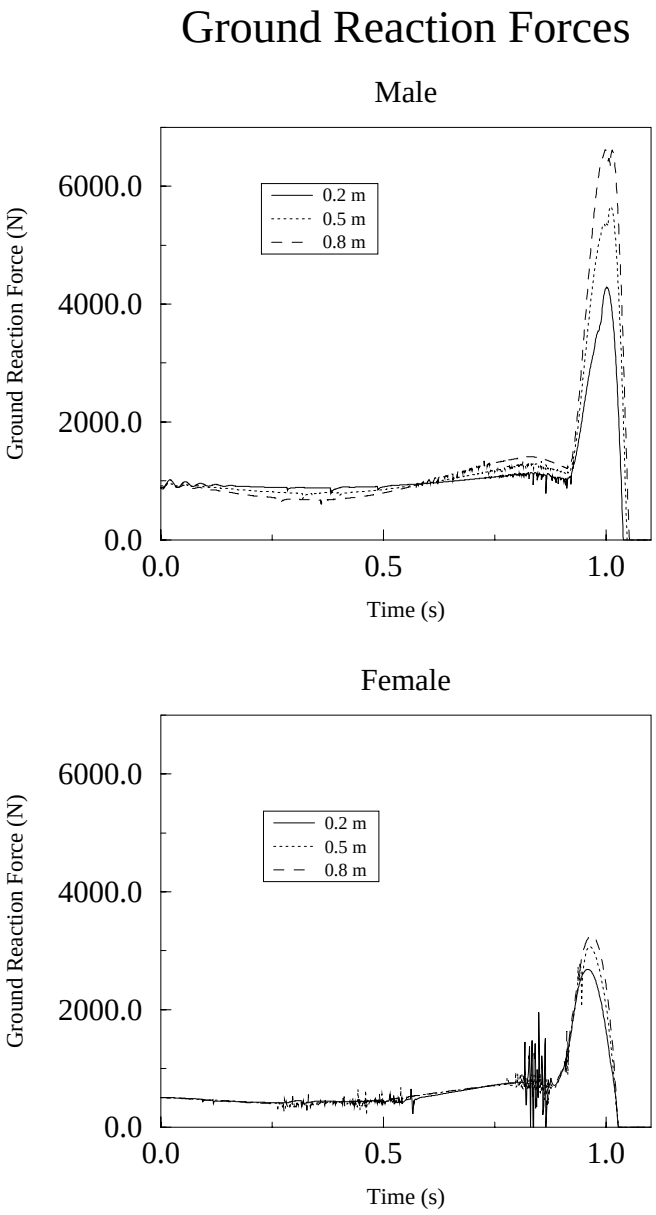


Figure 4.2. Ground reaction forces for leaps to 3 different heights. The noise in the reaction force of the female simulation is due to chattering in the ground contact model.

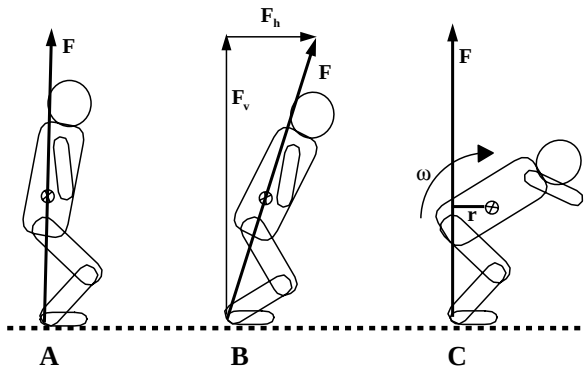


Figure 4.3. This figure conceptually illustrates the ground reaction force vector for (A) a vertical leap, (B) a broad jump, and (C) a leap with angular momentum. In the vertical leap, the ground reaction force F has no horizontal component. In the broad jump, F can be broken down into vertical and horizontal components F_v and F_h. In a leap that produces significant angular momentum, F does not pass through the center of mass and a torque acts on the body that increases ω during lift-off.

extending the lower limb joints) or fall down.

The leaping controller uses the desired horizontal distance specified by the user to choose values for the three lower-level parameters used in the standing broad jump (l_{hip}, Δ_{COM}, and $shld_{z_d}$). As in the vertical leap, smaller values of l_{hip} produce higher leaps. Higher values of Δ_{COM} increase the distance jumped. The desired shoulder angle, $shld_{z_d}$, causes the arms to swing forward as the lower limbs extend and slightly increases the horizontal velocity while mimicking the strategy used by humans. Table 4.2 shows the data used to interpolate the values of each parameter. The performance of the controller on leaps of various distances is shown in figure 4.4. In this case, three data points result in larger errors than those seen in the vertical leap. Finer resolution interpolation could be used to reduce the error.

The standing broad jump is also used to measure physical fitness. The Singapore Sports Council has devised a set of athletic performance measures, one of which includes the broad jump [Cou98]. The distance that a male in age group 25 to 34 should be able to jump is about 2.2 m. The distance a female in the same age group

Male

l_{hip}	Δ_{COM}	$shld_{d_z}$	COM distance
0.72	0.05	1.8	0.376
0.68	0.07	2.8	1.155
0.50	0.08	3.2	2.120

Female

l_{hip}	Δ_{COM}	$shld_{d_z}$	COM distance
0.68	0.02	1.8	0.31
0.52	0.04	2.8	0.69
0.40	0.06	3.2	1.57

Table 4.2. The distance the character's center of mass (COM) travels horizontally is used to select the three control parameters for the broad jump.

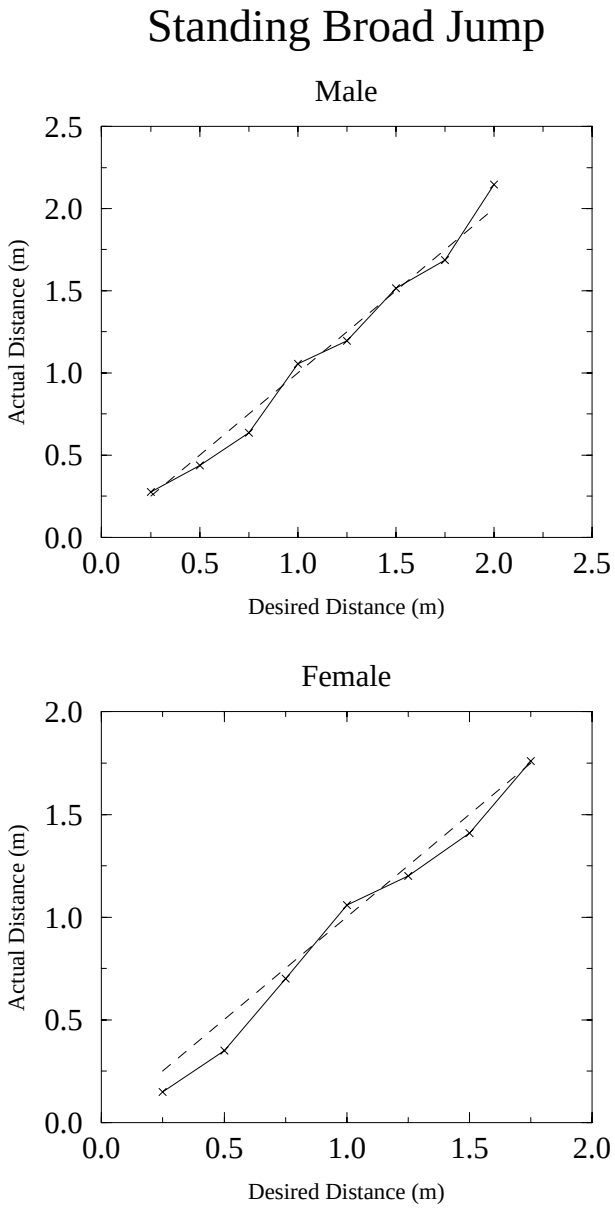


Figure 4.4. The length of the standing broad jump achieved by the character in response to a commanded distance. The dashed line represents a perfect performance where the achieved distance is equal to the desired distance.

should be able to jump is about 1.7 m. The maximum distance the male simulation can jump is 2.12 m and the maximum distance the female simulation can jump is 1.57 m. Both values are close to the average, but by increasing the gains of the servos, broad jumps of greater distances could be attained.

4.3 Angular Momentum

For some maneuvers, an athlete must generate substantial angular momentum. A successful forward somersault, for example, requires sufficient angular momentum to allow the athlete to complete a full revolution and land on his or her feet. The leaping controller generates angular momentum by modifying the ground reaction force vector.

The angular velocity of a body in space is related to the angular momentum,

$$\mathbf{H} = \mathbf{I}\omega \tag{4.1}$$

where $\mathbf{H}$ is angular momentum of the body, $\mathbf{I}$ is the moment of inertia and ω is angular velocity. Angular momentum can be added or removed only by an external torque acting on the body, therefore the angular momentum cannot be influenced while the body is in flight. The ground reaction force acting on the feet provides an external torque as the simulated character leaps into the air:

$$\tau = \mathbf{F} \times \mathbf{r} \tag{4.2}$$

where τ is the torque, $\mathbf{F}$ is the ground reaction force, and $\mathbf{r}$ is the line of action, which is perpendicular to the ground reaction force vector. Figure 4.3 shows how the torque is related to the cross product of the ground reaction force and the line of action from the character's center of mass. This torque (and correspondingly $\mathbf{H}$ and ω) can be increased by either the ground reaction force or the line of action.

The leaping controller uses both strategies. The ground reaction force is increased by increasing the gains of the servos in the hips, knees, and ankles just before the lower limbs are extended. However, the main method used to control angular velocity is by changing the length of the line of action. The controller does this by setting the desired hip angle at lift-off to a value that causes the ground reaction force to pass either behind or in front of the center of mass of the character. Figure 4.5 shows the angular velocity about the somersault axis attained when using different hip angles during lift-off.

4.4 Summary

The leaping controller described in this chapter can produce a range of vertical leaps and broad jumps. This controller can also alter angular momentum as the simulated character leaps into the air. The leaping controller could be generalized to include leaping sideways as well as vertically and horizontally. This leaping controller does not use torque limits at the joints. A more realistic approach would enforce torque limits to prevent the character from leaping to super-human heights.

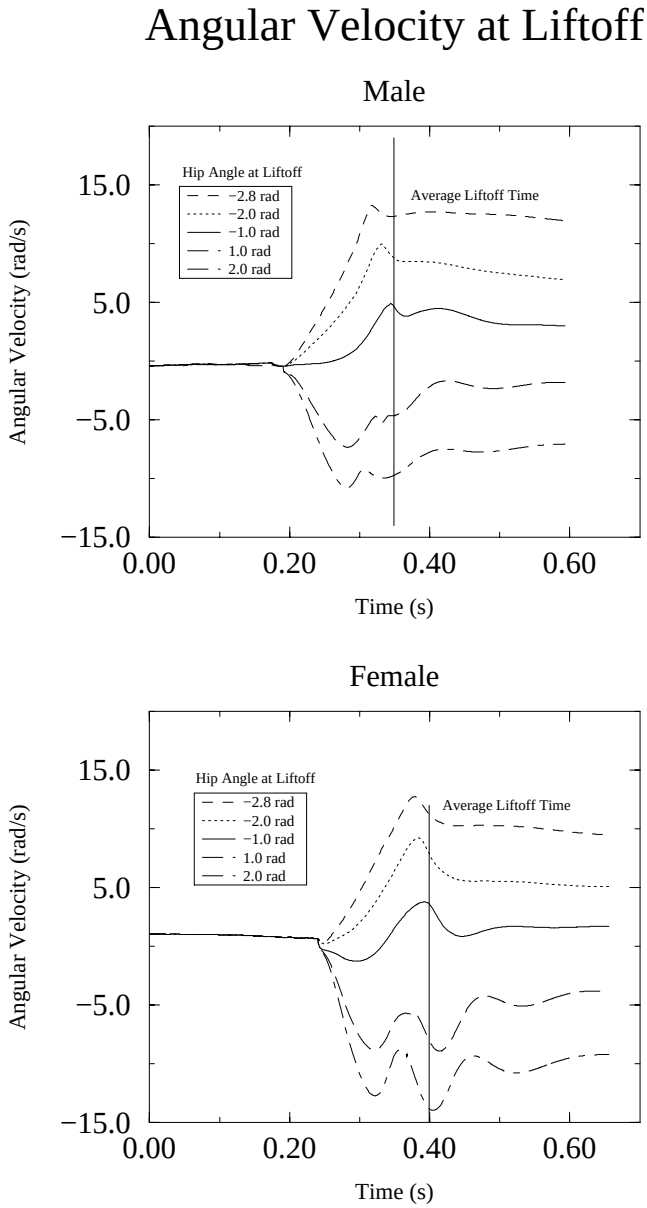


Figure 4.5. Angular velocity about the somersault axis (in the inertial frame) generated for 5 leaps. This experiment tested only the generation of angular velocity. In some trials the height of the center of mass was insufficient to perform a forward or backward somersault.

CHAPTER 5

Tumbling

This chapter describes the tumbling basis controller, which uses the angular momentum generated by the leaping controller to perform somersaulting and twisting maneuvers. The tumbling controller produces somersaults in a layout, pike, or tuck position and can initiate twists after the simulated character has become airborne. Figure 5.1 shows a simulated gymnast performing an aerial somersault and a simulated diver performing a twist. A description and analysis of the somersaulting controller is provided in the first section and the second section describes and analyzes the controller for twisting.

5.1 Somersaults

Athletes normally perform somersaults using the angular momentum generated as they leap into the air. Once airborne, the style and number of somersaults they intend to perform determines whether they enter a layout (no flexion), a pike (flexion at the hips) or a tuck (flexion at hips and knees) (figure 5.2). The tumbling controller initiates somersaults in a similar fashion by using the leaping controller to generate angular momentum and then positioning the body into a layout, pike, or tuck.

Athletes can alter their moment of inertia about the somersaulting axis by changing their body position. Because angular momentum is conserved, a lower moment of inertia will result in a higher angular velocity. Tighter tucks are often used by athletes to achieve a greater number of somersaults. For example, the female character

Figure 5.1. A simulated male gymnast performing a somersault and a simulated female diver performing a twist.

reduces her moment of inertia about the somersaulting axis by a factor of three when she moves from a layout to a tuck position. Figure 5.2 gives the moment of inertia about the somersaulting axis for the male and female characters for the pike, tuck, and layout positions. Using predetermined angles, the tumbling controller flexes the hips and spine to cause the character to enter a pike position, or flexes the knees, hips, and spine to enter a tuck position. Figure 5.3 illustrates the increase in angular velocity as the male and female characters move from a layout to a pike and tuck position.

5.1.1 Torque-Free Somersault

Even if the character has no angular momentum, a limited amount of body rotation about the somersaulting axis can be accomplished because the moment of inertia of various body parts about the somersaulting axis are not equal. For example, if the male character windmilled the arms clockwise 360 degrees, the torso would tilt counter-clockwise by 20 degrees because the moment of inertia of the arms is much smaller than the moment of inertia of the body. However, if the arms are then

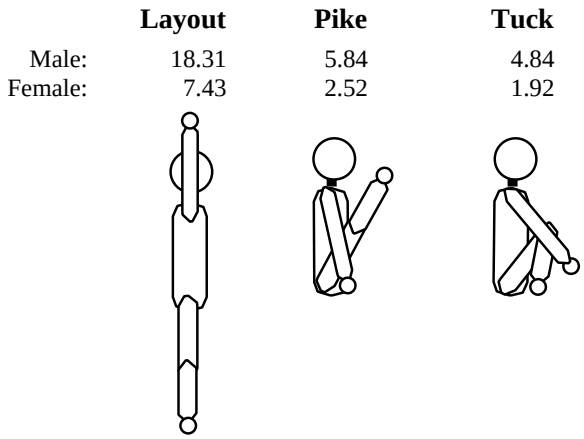


Figure 5.2. Three different positions used during a somersault and corresponding moments of inertia (in kgm^2) for the male and female character about the somersaulting axis.

windmilled back counter-clockwise by 360 degrees, the torso will return to its original position because angular momentum is conserved.

An athlete can seemingly violate the law of conservation of momentum and perform a partial somersault of nearly 90 degrees by performing a "backdrop" maneuver [Fro79]. A series of limb motions (figure 5.4) take advantage of the differences in moments of inertia to perform a torque-free half "somersault". The limb motions perform a set rotations that can be undone in such a way that the athlete does not return to the original vertical position. By repeating this sequence of limb motions several times, the simulation can perform a complete somersault, as could a human given a sufficiently long flight time.

5.2 Twisting

Divers and gymnasts often combine twists with somersaults to increase the difficulty of a maneuver and create a routine that is visually dramatic. Twists with high angular velocities can be produced in the layout position, because the moment of inertia about

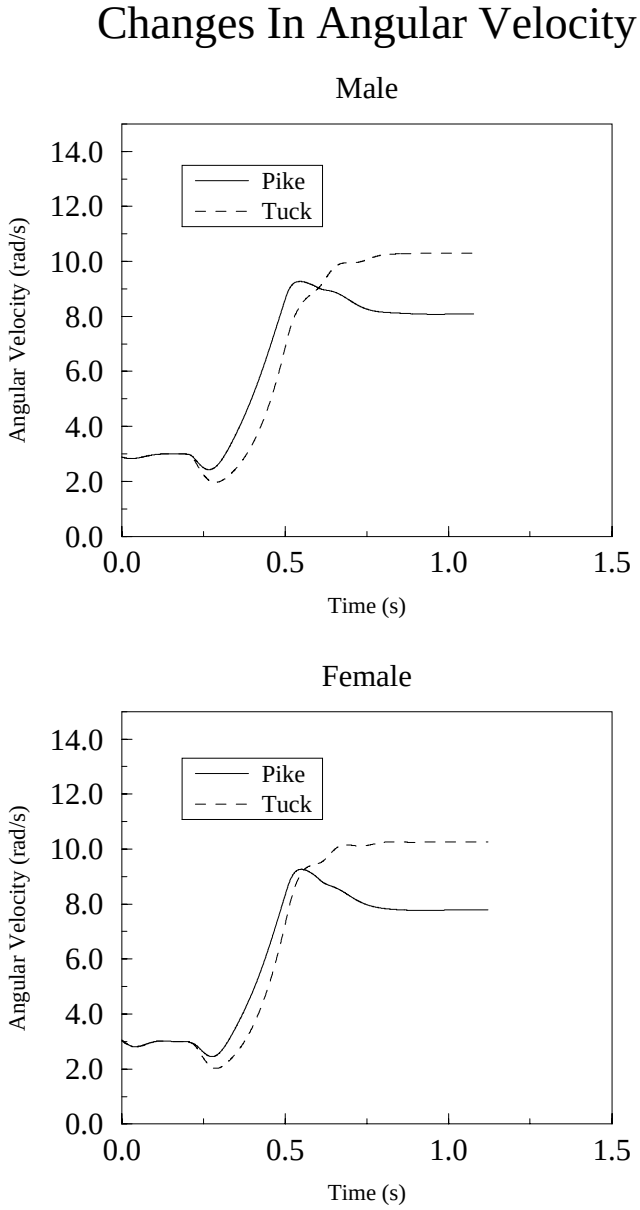


Figure 5.3. Changes in angular velocity about the somersaulting axis as the character moves from a layout to a pike or tuck position. The decrease in angular velocity in the pike position results when the controller overshoots the desired hip angles.

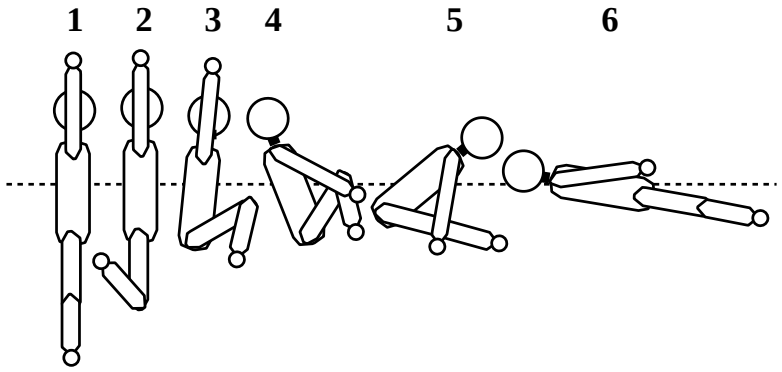


Figure 5.4. This diagram illustrates the sequence of limb motions required to perform a partial somersault in the absence of angular momentum. Figure adapted from Frohlich [Fro79].

the twisting axis is small. The most remarkable types of twist are those in which the athlete initiates a twist in the air with no apparent velocity about the twisting axis by throwing one arm behind the head and the other across the torso while performing a somersault. The clockwise motion of the arms causes a counter-clockwise rotation of the torso and the athlete's body is no longer aligned with the somersaulting axis (figure 5.5). The athlete then begins to rotate about the twisting axis while rotating about the somersaulting axis with a slightly reduced angular velocity.

The creation of this type of "torque-free" twist [Fro79] can be analyzed more closely by examining the athlete's moment of inertia tensor $\mathbf{I}$ in an inertial reference frame. When the athlete initiates the maneuver, $\mathbf{I}$ contains no off-axis components. The athlete's principal axes are aligned with the axes of the inertial reference frame. The angular momentum $\mathbf{H}$ and angular velocity ω are also aligned (figure 5.6). After the arms are thrown, the principle axes are no longer aligned with the axes of the inertial reference frame. Therefore ω must possess non-zero components along other axes to ensure that $\mathbf{I}\omega = \mathbf{H}$ remains constant in the inertial reference frame.

The tumbling control system initiates a twist by throwing the arms as described above. To produce multiple twists, a larger somersaulting angular velocity is gener-

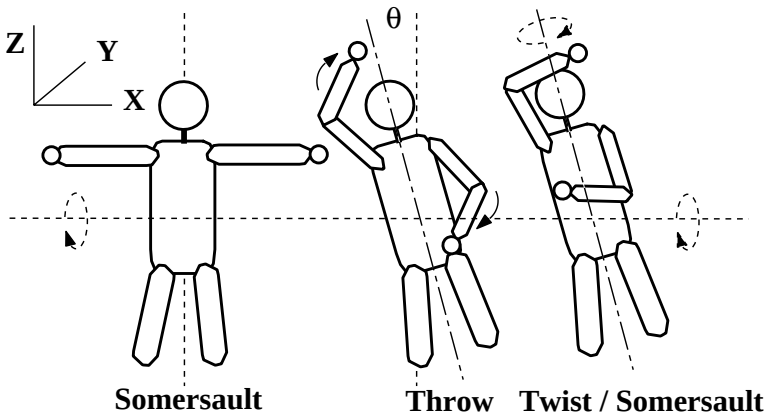


Figure 5.5. When an athlete in a layout position throws the arms while performing a somersault, a twist is initiated.

ated before the arms are thrown. Analysis of the angular velocity about the rotational axes of the male and female characters demonstrates that this method is successful. The graphs in figure 5.7 show the angular velocity in the athlete's reference frame just after the arms have been thrown. The angular velocity about the Z axis (the twisting axis in this case) increases after the arms are thrown. At time 0.728 s (marked by the vertical bar) the arms are thrown back to the sides to undo the twist.

5.3 Summary

This chapter presented a tumbling controller for generating combinations of twists and somersaults. Transitions from one tumbling maneuver to another occured based on timing information. Future versions of the controller might transition between aerial maneuvers based on altitude or angular velocity. Future versions of the controller might also control angular velocity while in flight by increasing or decreasing the tightness of a pike or tuck.

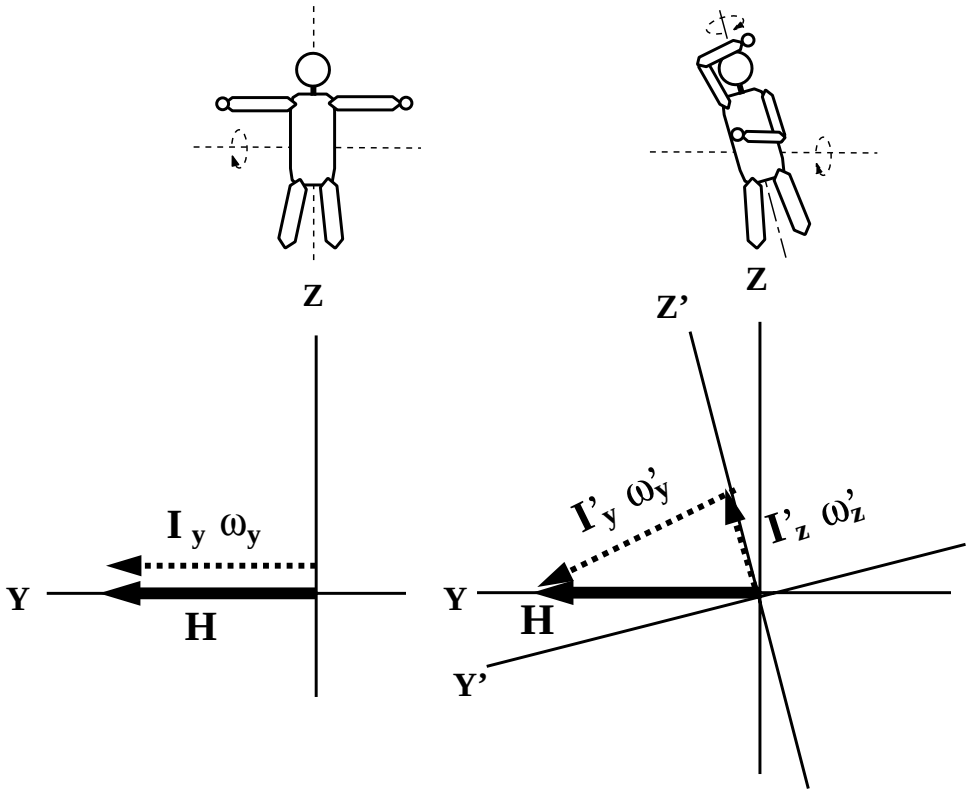


Figure 5.6. Vector diagrams of the angular momentum of an athlete before and after the arms have been thrown. Before the arms are thrown, the athlete's body is aligned with the twisting axis (Z). Afterwards, the athlete's body is no longer aligned and there is a component of angular velocity about the (Z') axis in order to ensure that **H** does not change.

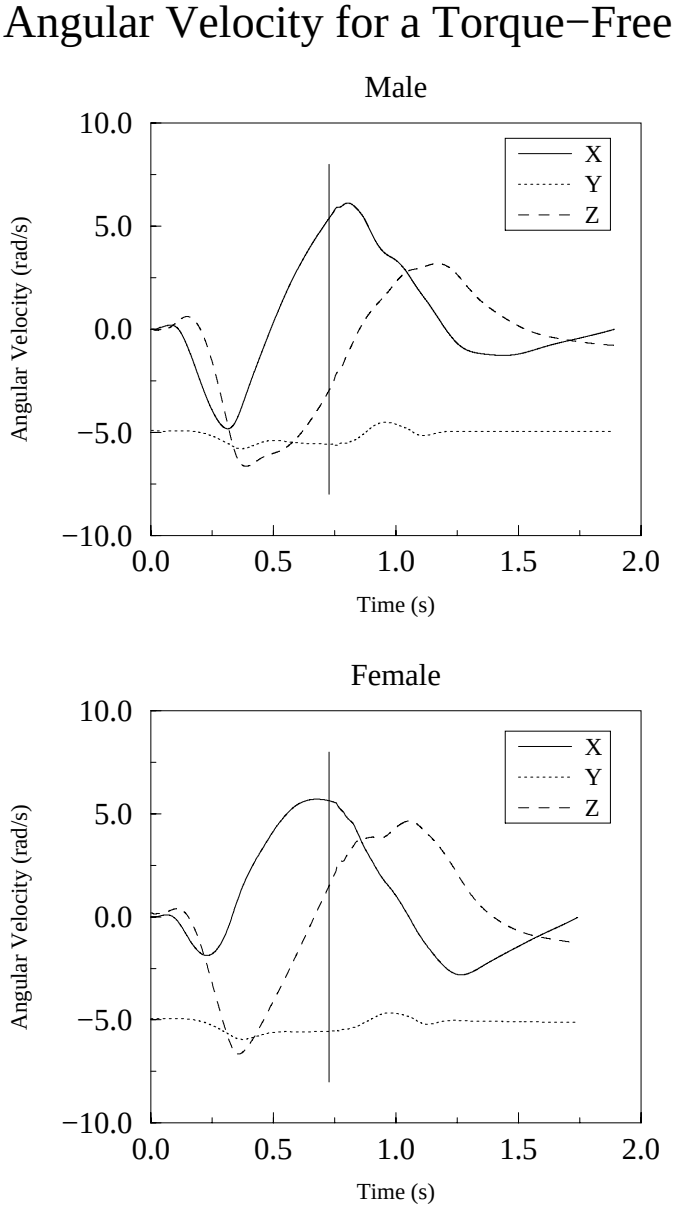


Figure 5.7. This graph shows the change in angular velocity about the X, Y, and Z axis (in the athlete's reference frame) as the character moves its arms as shown in figure 5.5. Angular velocity exists about the X and Z axis because the character is somersaulting as well as twisting.

CHAPTER 6

Landing

The previous chapter described a basis controller for performing aerial maneuvers when the character was in free-fall. This chapter describes the landing basis controller that takes the character from an aerial state to a balanced state by reducing the vertical, horizontal, and angular velocities.

The landing controller is most effective at producing landings in which the character does not need to reposition the feet after touchdown. This controller can land from vertical drops as high as 1.15 m and with horizontal velocities between 3.0 m/s and -1.0 m/s. The first section of this chapter explains how the controller lands from vertical drops with and without horizontal velocity. The second section analyzes and evaluates the performance of the control system.

6.1 Passive Landings

When an athlete lands on the ground, he or she must dissipate most of the translational and rotational energy of the maneuver. In the process of removing this energy, the linear and rotational velocities of the center of mass are reduced sufficiently so the athlete is able to establish an upright, balanced posture. To maintain a static balanced posture, the vertical projection of the athlete's center of mass onto the ground plane must lie within the area of support defined by the feet (after the horizontal velocity is dissipated). If the athlete has significant horizontal velocity before landing, he or she will not be in a balanced state at touchdown, because the feet will land

forward of the projected center of mass. If the feet land too close to the center of mass, there is not enough time to reduce the horizontal velocity and the athlete will fall over forward. If the feet land too far in front of the center of mass, the horizontal velocity will be reduced to zero before the center of mass falls within the area of support and the athlete will fall over backwards (figure 6.1).

The landing controller uses passive control (as opposed to actively servoing the ankles) to bring the center of mass over the area of support. The joint angles of hips, knees, and ankles at touchdown are calculated to place the feet such that the simulated character's forward velocity will be near zero when the center of mass falls within the area of the support. The hip angles are computed by adding an offset γ_x to the desired hip angles. The offset is computed by a proportional-derivative servo:

$$\gamma_x = k_p(Feet_{x_d} - Feet_x) - k_v \dot{Feet}_x \tag{6.1}$$

where k_p and k_v are the gains, $Feet_x$ and $\dot{Feet}_x$ are the position and velocity of the center of mass of both feet in the character's local coordinate space, and $Feet_{x_d}$ is the desired location for the feet. The desired ankle angles are computed so that the feet are parallel to the ground and the knees are flexed to absorb the shock of landing.

The exact placement of the feet that results in a successful landing is controlled by adding a correction term, called the foot servo delta, to the hip offset servo (γ_x). This foot servo delta parameter was interpolated from a number of trials at various horizontal velocities (table 6.1). After touchdown, the position gains on the knees are set to zero and the damping gains are increased. The feet are servoed to keep them flat on the ground, and the hips are servoed to keep the pelvis vertical while the knees bend. After the angular velocity of the knees has been reduced to 20 percent of the angular velocity at touchdown, the knee gains are restored to their original values. The controller transitions to balance when the center of mass moves past the desired center of mass, has a very low velocity near the desired center of mass, or starts to

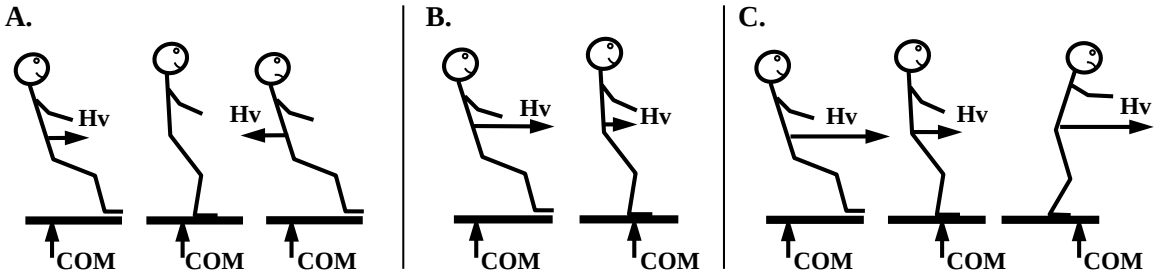


Figure 6.1. If the feet are too far forward of the center of mass (COM), the character will fall over backwards (A). If they are too far behind the COM the character will fall forward (C). If the feet are positioned correctly the character will achieve balance (B). H_v represents the horizontal velocity of the character.

Figure 6.2. The feet of the male character are placed (using the hip joints) forward of the center of mass during a landing from a broad jump. The knees are also bent to absorb energy during the landing.

COM Horiz velocity (m/s)	Foot Servo Delta	
	Male	Female
3.0	0.16	-0.06
2.0	0.20	0.04
1.0	0.20	0.14
0.0	0.08	0.08
-1.0	-0.01	-0.08

Table 6.1. A correction term for the foot position servo is interpolated to achieve a successful landing and allow the character to transition to the balance controller.

move away from the desired center of mass. Figure 6.2 shows an example of the foot location relative to the center of mass as the male character lands.

6.2 Evaluation

This section presents the results of three experiments used to assess the performance of the landing controller. The first test evaluated the controller's performance for vertical drop landings and the second evaluated performance for landings with horizontal velocity. The third experiment was similar to the second, except additional mass was added to the female character.

The first experiment determined how far the character's center of mass could fall while still being able to recover balance. The character's center of mass was positioned vertically above the ground with no horizontal velocity and was allowed to fall under the influence of gravity. The height was initially increased by 0.5 m increments and then by 0.01 m increments near the failure region. The male character was able to recover from a fall of 1.15 m and the female character from a fall of 0.92 m.

The second experiment evaluated the maximum horizontal velocity that the control system could tolerate. In this test, the center of mass was raised to a height

COM Horiz velocity (m/s)	Foot Servo Delta 2 kg	4 kg
3.0	-0.06	-0.06
2.0	0.04	0.06
1.0	0.14	0.15
0.0	0.08	0.08
-1.0	-0.085	-0.105

Table 6.2. To allow landings with added mass, the foot servo delta had to be modified from the 0.0 kg case. With 2.0 kg of additional mass, the controller needed modification only at -1.0 m/s. With 4.0 kg of additional mass, the controller required modification at all velocities except 0.0 m/s.

of 1.25 m and the initial forward velocity of the center of mass was increased until the character could not land successfully. The landing controller for each character was able to land with horizontal velocities between 3.0 m/s and -1.0 m/s. Figure 6.3 illustrates the effect of foot position on the horizontal velocity during landing.

The third experiment added a 2.0 kg and 4.0 kg backpack to the female and determined the changes that needed to be made to the foot servo deltas while landing under the same conditions as the previous experiment. No other control parameters were modified and the results of the experiment can be see in table 6.2.

6.3 Summary

The landing controller described in this chapter allows the simulated character to transition from an airborne state to a standing, balanced state. A more general controller for landing would allow the character to land and if necessary take a step to regain balance. The controller could also be generalized further to allow the character to land with significant lateral horizontal velocities.

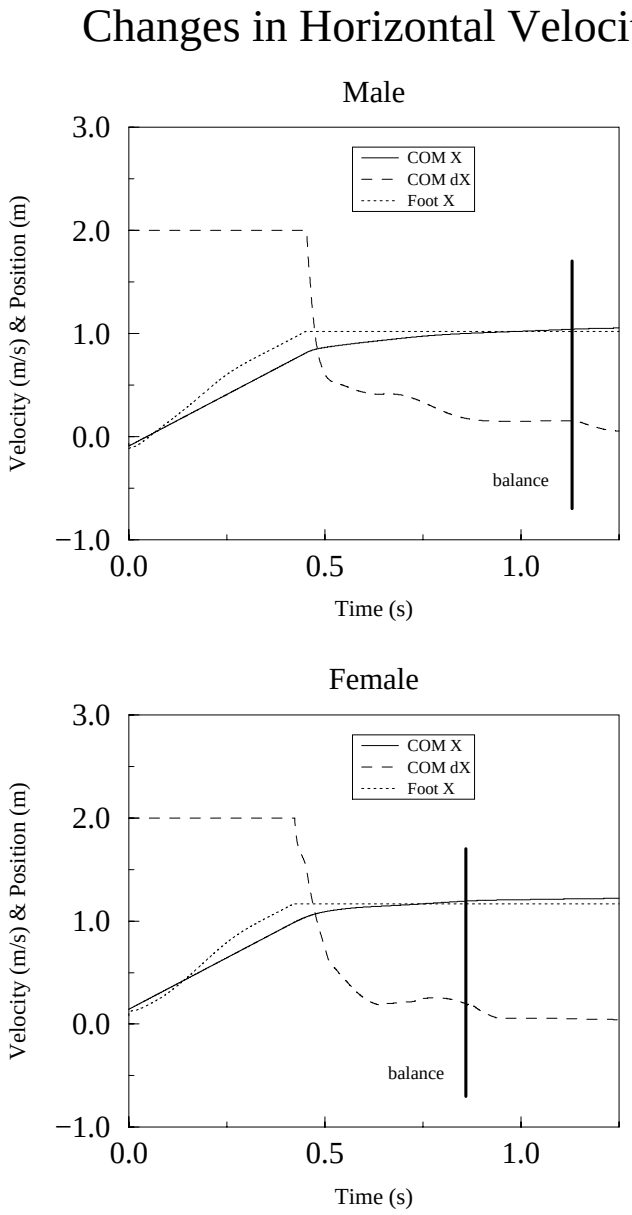


Figure 6.3. The velocity drops significantly when the character contacts the ground. The feet are placed ahead of the center of mass and the velocity drops to near zero as the center of mass travels forward over the feet. The vertical bar indicates when the balance controller is activated.

CHAPTER 7

Balancing

The balance basis controller is used to maintain an upright, standing posture for the male and female model. The controller allows the user to specify trajectories for joints in the upper torso without requiring specification of the corresponding hip, knee, and ankle angles that will maintain balance. This basis controller gives the user control over the pitch of the torso, the height of the hips above the ground, and the location of the character's projected center of mass. The next section provides implementation details for this controller, section 2 discusses the user control parameters, and the last section evaluates the performance.

7.1 Maintaining Balance

An athlete maintains stability as long as the vertical projection of his or her center of mass onto the ground lies within their area of support. The athlete maximizes stability by actuating the ankles and hips to keep their projected center of mass near the center of their area of support (figure 7.1).

The balance controller maintains stability in a similar fashion by keeping the center of mass of the simulated character near the center of the area of support. The controller determines the perimeter of the area of support by computing the two-dimensional convex hull of the contact points of both feet. Using the center of the area of support (taken to lie between the extremes of the convex hull along the X and Y axis) and the position and velocity of the projected center of mass, two

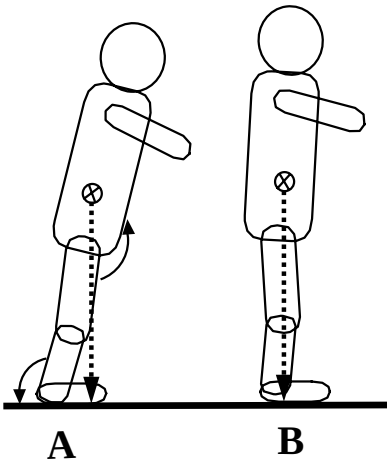


Figure 7.1. By actuating the hips and ankles (A), the character can keep the projected center of mass near the center of the area of support (B).

proportional-derivative servos (one for the X axis and one for the Y axis) compute offsets δ_x and δ_y to the desired angles of the hip and ankle servos:

$$\delta_x = k_{p_x}(COM_{x_d} - COM_x) - k_{v_x}C\dot{O}M_x \tag{7.1}$$

$$\delta_y = k_{p_y}(COM_{y_d} - COM_y) - k_{v_y}C\dot{O}M_y \tag{7.2}$$

where k_{p_x}, k_{p_y}, k_{v_x}, and k_{v_y} are the gains of the proportional-derivative servo, COM_x, COM_y, $C\dot{O}M_x$, and $C\dot{O}M_y$ are the positions and velocities of the character's center of mass in the character's local coordinate space, and COM_{x_d} and COM_{y_d} are the desired locations for the center of mass. The offsets are added to the current desired servo positions for the ankle and hip. By using δ_x and δ_y as offsets, nominal ankle and hip angles can be specified by other control processes while the balance controller attempts to maintain balance. The offsets for the hips are proportional to the offsets for the ankles but have the opposite sign. In this case the hip offsets were equal to half the ankle offsets.

During balance the ankles keep the feet flat on the ground:

$$ankle_{x_d} = ankle_x - foot_\rho \tag{7.3}$$

$$ankle_{y_d} = ankle_y - foot_\theta \tag{7.4}$$

where $ankle_{x_d}$ and $ankle_{y_d}$ are the desired positions for the ankle servos about the X and Y axis, $ankle_x$ and $ankle_y$ are the current orientations of the ankle joint about the X and Y axis, and $foot_\rho$ and $foot_\theta$ are the roll and pitch of the foot about its center of mass. Joint limits prevent the ankles from extending beyond 90 degrees or flexing beyond -68 degrees.

The balance controller operates in the local coordinate space of the character (with the front of the pelvis facing the positive X axis and the left side of the pelvis facing the positive Y axis). By using this coordinate system, the balance controller maintains stability regardless of the orientation or location of the character in world space. A limitation to this method is that the performance of the control system degrades when the positive X axis of the pelvis is not aligned with the positive X axis of the character's feet. However, large angles between the pelvis and feet do not normally occur when balancing on both feet.

7.2 User Control Parameters

Three parameters provide control over the motion of the simulated character while the balance controller is active: torso pitch, pelvis height, and a center of mass offset. The pitch of the upper torso is used for behaviors where the simulated character needs to lean forward or backward. For example, this parameter allows the female character to bend over and touch her toes (figure 7.2).

Controlling the height of the pelvis above the ground plane allows the character to do kneebends to various heights. The control system servos the knee angle using the kinematics of the legs:

$$knee_{y_d} = \pi - acos\left(\frac{ul^2 + ll^2 - l_{hip}^2}{2\ ul\ ll}\right) \tag{7.5}$$

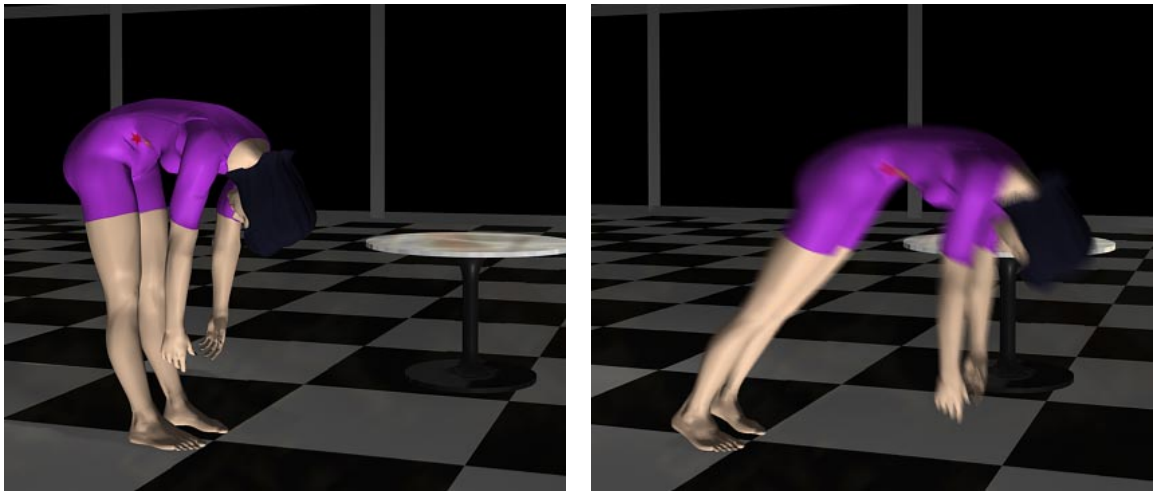


Figure 7.2. The pitch of the upper body can be specified to produce behaviors where the character bends over. The two characters have the same desired pitch of the upper body but the balance controller is not active for the character on the right and she falls over.

where $knee_{y_d}$ is the desired knee angle, ul is the length of the upper leg, ll is the length of the lower leg, and l_{hip} is the desired crouch height. The hip joint keeps the pelvis vertical:

$$hip_{y_d} = -acos\left(\frac{ul^2 - ll^2 + l_{hip}^2}{2\ ul\ l_{hip}}\right) \tag{7.6}$$

where hip_{y_d} is the desired hip angle. Figure 7.3 provides an illustration of the terms used in the equations above. Figure 7.4 illustrates the effect of a change in hip height on the performance of the balance controller for both the male and female simulations.

An offset to the desired center of mass (COM_{x_d} and COM_{y_d} in equations 7.1 and 7.2) allows the user to alter the character's balance. If the projected center of mass falls outside the area of support the character will no longer be able to maintain balance and will have to leap into the air or fall down. This parameter is used to move the projected center of mass when transitioning to the leaping controller.

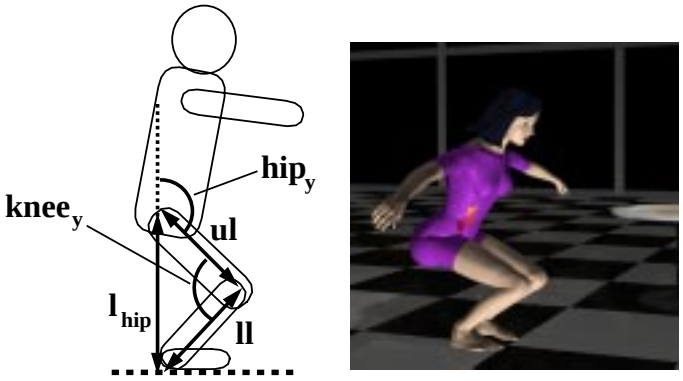


Figure 7.3. The terms used in the law of cosines for computing the desired knee angle to move the hip to the specified height above the ground.

7.3 How Well is Balance Maintained?

Four experiments were used to assess the performance of the balance basis controller. The experiments measured maximum offsets to the projected center of mass (with and without additional mass), maximum force disturbances to the torso, and maximum ground slopes.

The first experiment determined how far the center of mass could deviate from the center of the area of support before balance failed. An offset to the desired center of mass was specified in 8 directions. A search for the largest offset that allowed balance to be maintained (by keeping the center of mass with the support polygon) was performed in 0.005 meter increments. Figure 7.5 shows the results of the experiment. The actual center of mass overshoots the desired center of mass by a significant amount for both the male and female model. Performance is better along the X axis than the Y axis because one of the feet lift off the ground if the center of mass is offset too far in the Y direction. A more robust control system would shorten one of the legs to prevent a foot from leaving the ground or continue to maintain balance even if one of the feet left the ground.

The second experiment was identical to the first except that the mass parame-

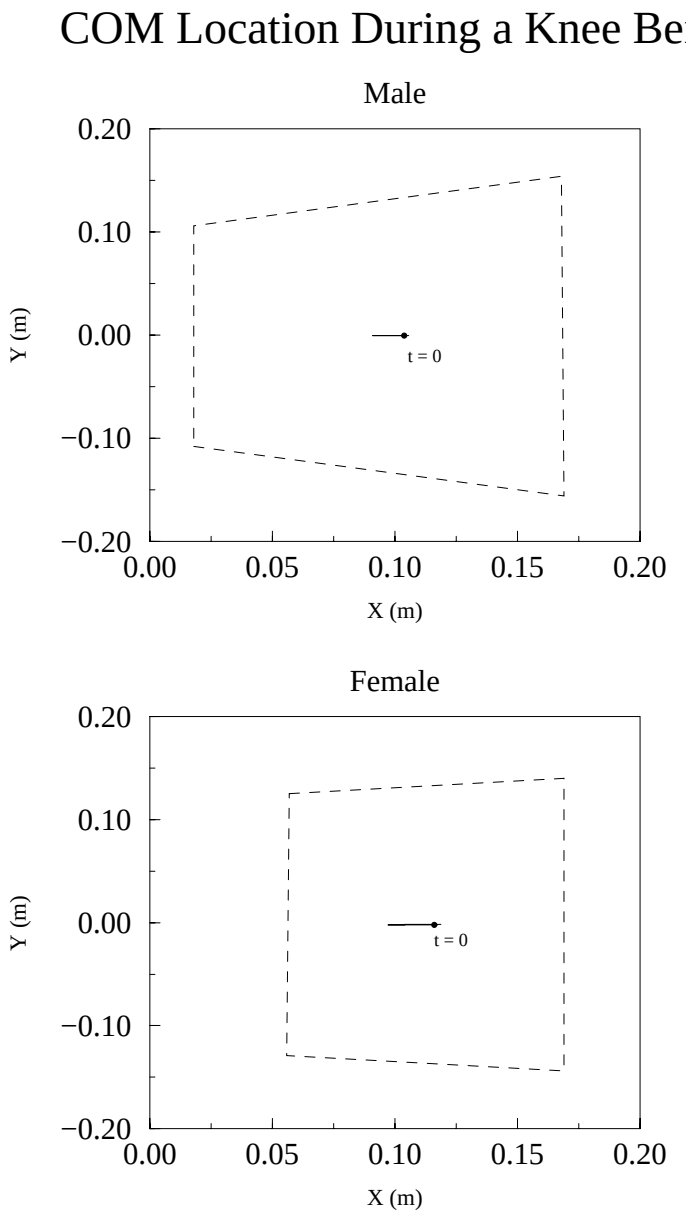


Figure 7.4. The two graphs show the location of the projected center of mass (the solid line) as the male model moves his hips to a height of 0.52 meters and the female model moves hers to a height of 0.47 meters. The perimeter of the area of support is designated by the dashed line (the male has larger feet, hence the larger area of support). The initial location of the center of mass is marked with a dot at t=0.

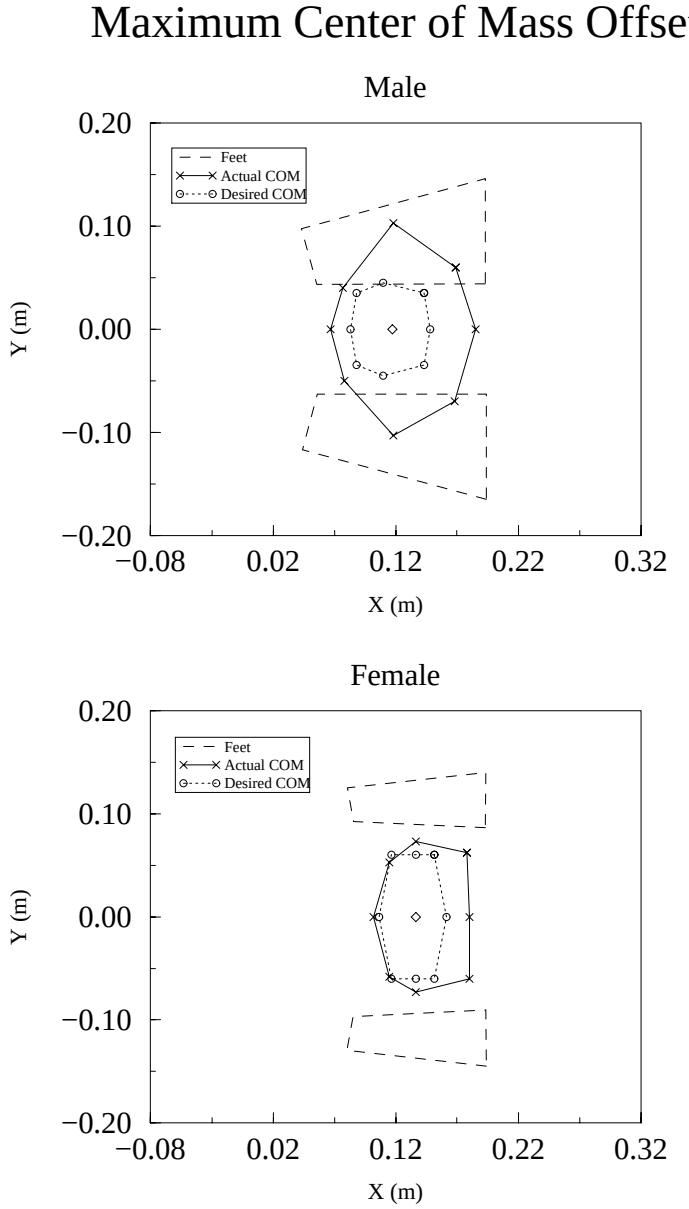


Figure 7.5. The graphs show the maximum offsets to the center of mass that still allow balance. The actual location of the center of mass was recorded when the maximum offset from the center of the area of support was found. The diamond indicates the center of the area of support.

ters of the female model were modified. Figure 7.6 shows the results from two female simulations, one wearing a 2.0 kilogram backpack and the other wearing a 4.0 kilogram backpack. All control parameters were the same as the original simulation. The addition of a 2.0 kilogram backpack did not alter the behavior of the controller significantly. The 4.0 kilogram backpack caused the offsets to the desired center of mass to be smaller, but the actual location of the center of mass was similar to the test with the 2.0 kilogram backpack.

A third experiment examined the balance controller's ability to recover from external disturbances. In this experiment, eight forces were applied from different directions to the center of mass of the sternum for 0.25 s. Each force was incremented by 5.0 N until the maximum force that still allowed balance was found. Figure 7.7 shows the results of the experiment. The male simulation recovers from larger disturbances than the female character, however with larger servo gains, the female simulation could probably recover from larger disturbances. The characters tended to fail more easily with forces along the Y axis because a foot lifts off the ground and causes the control system to fail.

The final experiment analyzed the ability of the balance controller for the male simulation to maintain balance on a sloped ground plane. The control parameters were the same as the ones used for balance on level ground. The slope of the ground plane was changed from a flat level surface to the maximum slope under which balance could be maintained. The offset to the desired center of mass was empirically changed over time to help maintain balance. The maximum slope in the $+X$ direction was 33.7 degrees, in the $-X$ direction 38.6 degrees, and in the $\pm$Y direction 25.7 degrees.

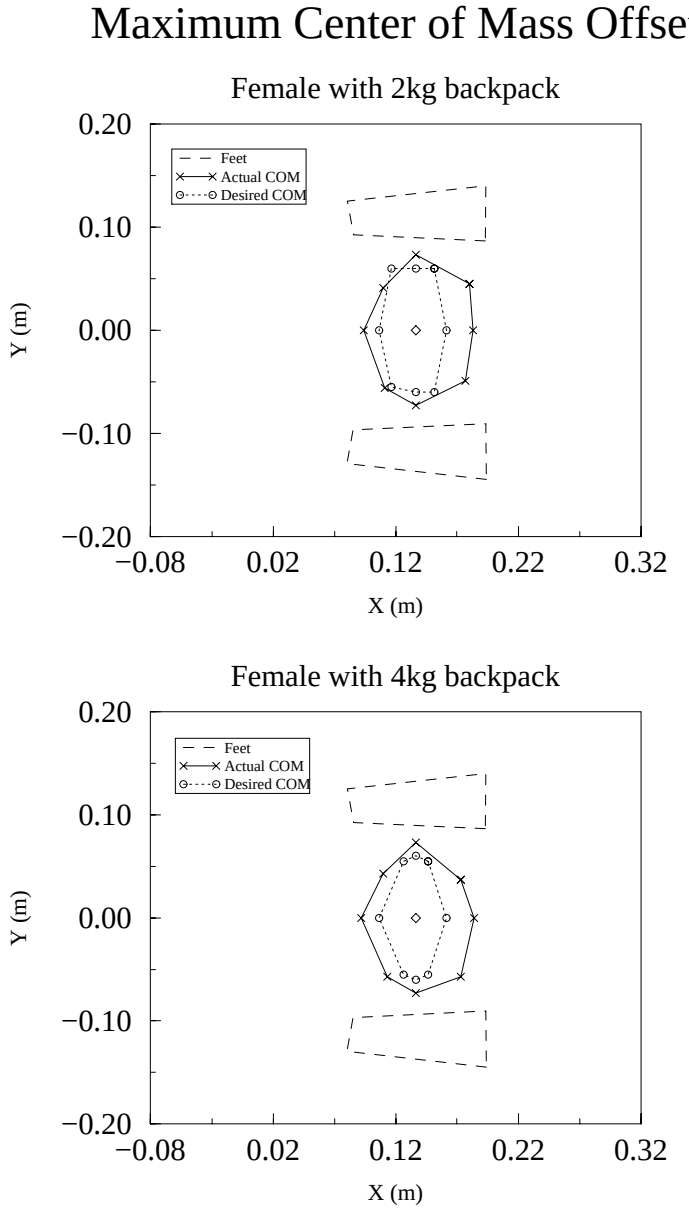


Figure 7.6. The graphs show the maximum offsets to the center of mass that allow balance to be maintained after mass has been added to the character. In both cases the control parameters were the same as in the female simulation shown in figure 7.5.

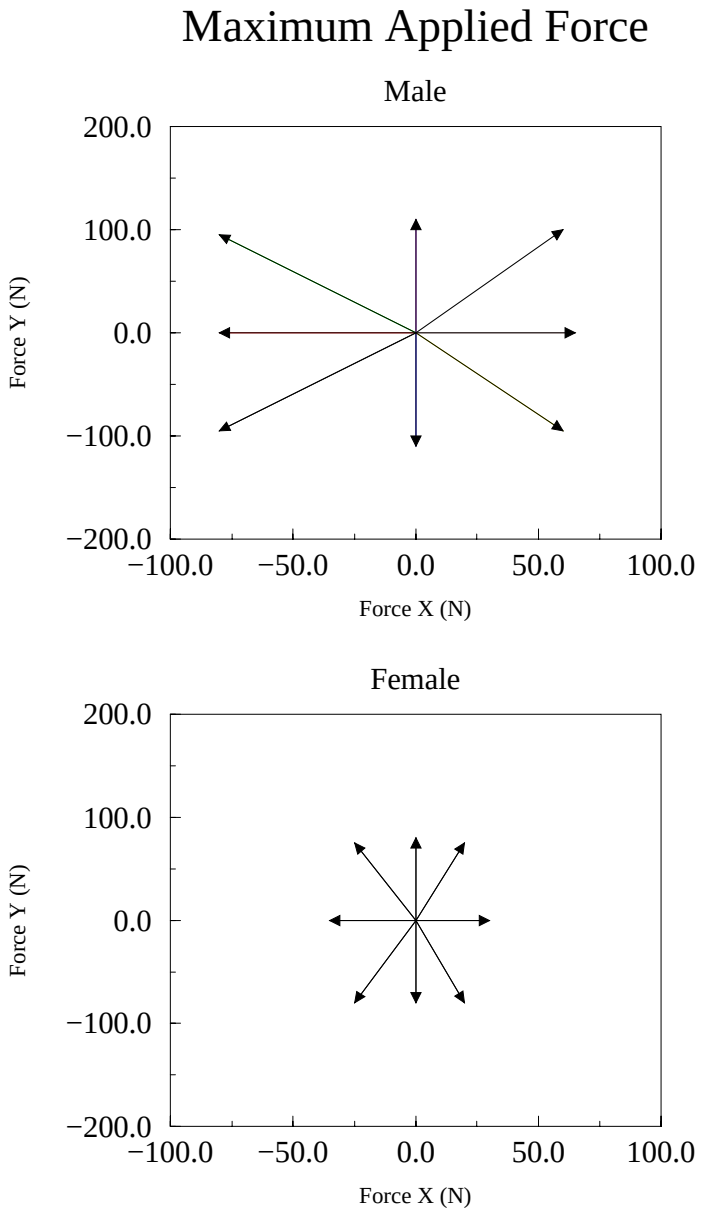


Figure 7.7. The graphs show the maximum force that could be applied to the center of mass of the sternum of the character over a 0.25 second interval. The characters were facing the positive X axis for this test.

7.4 Summary

This chapter described a balance controller that allowed simulated characters to maintain balance while standing on two feet. This controller allowed both characters to perform knee bends, as well as move upper body segments while maintaining balance. A more robust balance controller would allow the character to maintain balance on one foot and would allow the character to reposition the feet.

CHAPTER 8

Transitions

The four previous chapters described individual basis controllers, but the power of the approach taken in this thesis comes through combining the controllers to produce complex and varied behaviors. A wide variety of complex behaviors can be created by combining relatively simple basis behaviors. Each basis controller described in the previous chapters was simpler to design than an equivalent special purpose controller for the complex tasks described in this chapter. For example, the basis controller for balance required less effort to design than a single controller that could perform a forward somersault. A more general system results from breaking down complex behaviors into their component parts. By combining basis controllers many different behaviors can be produced, whereas a single, specialized controller generates a limited set of behaviors. However, transitions between basis behaviors must occur easily otherwise the user will spend most of his or her effort attempting to concatenate basis behaviors.

To generate transitions in a fairly automatic fashion, I have designed each basis controller to have a large state-space capture region, the set of initial conditions for which the controller can perform the desired behavior. If the capture region is sufficiently large, then the final state of each basis controller will leave the character in a valid initial state for the next controller and transitions between the behaviors will be automatic. To demonstrate the success of this approach, I describe a number of behaviors that were produced by combining the leaping, tumbling, landing, and balance controllers.

8.1 Vertical Leap

One of the simplest behaviors that can be created by combining basis controllers is the vertical leap where the character jumps into the air, lands on the ground, and maintains balance. This maneuver requires transitions from balance to leaping to landing and back to balance. Figure 8.1 illustrates the female character performing vertical leaps to three different heights.

Although there are many ways in which the human and simulated vertical leaping motion are similar (figure 8.2), the arms of the simulated character accelerate abruptly and do not travel through the same range of motion as do the arms of the human subject. However, the arm motion could be tuned to produce a more natural looking vertical leap by increasing the amount of time it takes for the arms to reach their desired positions and by making the arms more vertical at the apex of the leap.

8.2 Broad Jump

The broad jump is similar to the vertical leap, except that the leaping controller produces a horizontal displacement of the center of mass and the landing controller must dissipate the horizontal velocity at touchdown. As the character transitions from balance to leaping, the center of mass is positioned forward of the area of support. Then as the character's feet push off the ground, the ground reaction force provides a horizontal displacement to the character's center of mass.

The landing controller is activated as soon as the feet leave the ground in the broad jump. In the vertical leap, the landing controller is not activated until after the character reaches the apex of the leap. The earlier transition is required to allow sufficient time to bring the feet forward in preparation for landing. Figure 8.3 illustrates the male character performing a broad jump.

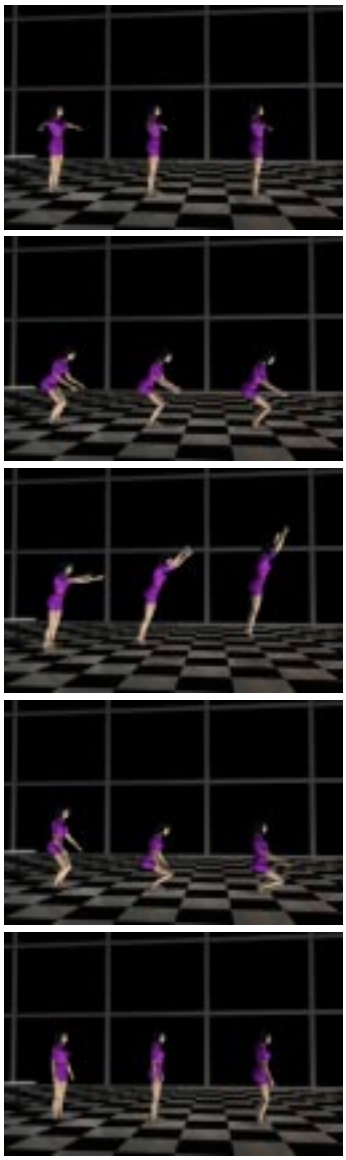


Figure 8.1. Filmstrip demonstrating vertical leaps to three different heights (0.16 m, 0.33 m, and 0.58 m) for the female character. The time interval between frames is 0.66 s.

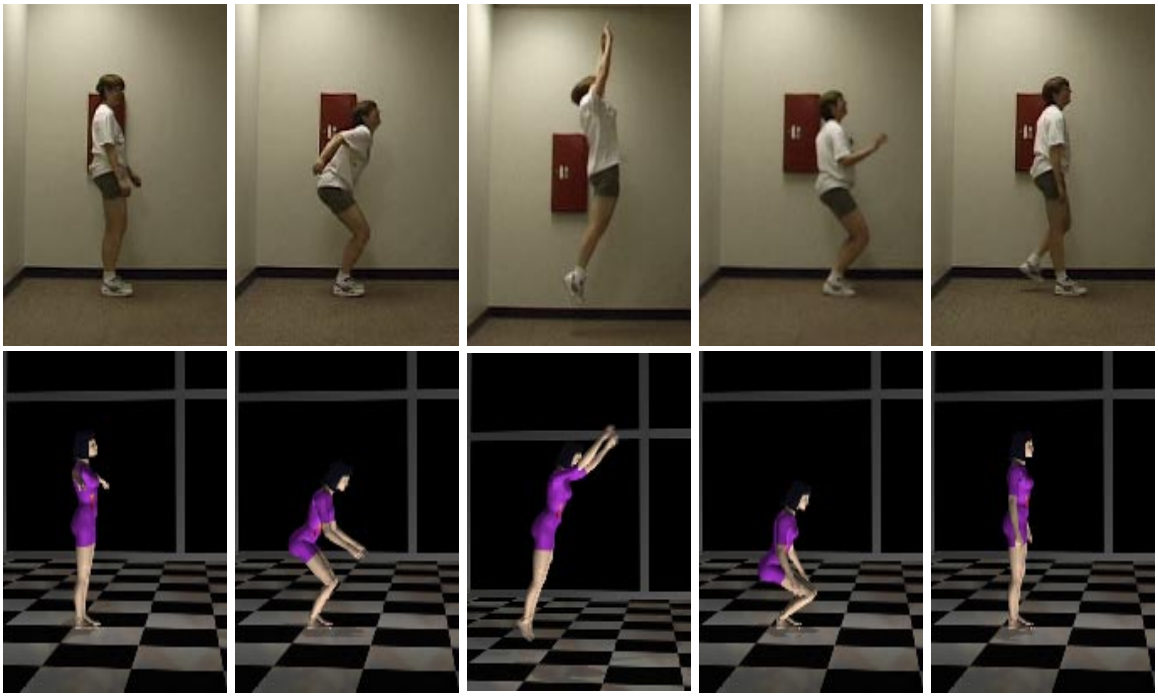


Figure 8.2. Comparison of human vertical leap (top) and simulated vertical leap (bottom). The time interval between frames is 0.33 s.

Figure 8.3. Filmstrip demonstrating the male character performing a broad jump to a distance of 1.1 m. The time interval between frames is 0.66 s.

A comparison between the simulated character and a human performance (figure 8.4) shows that the simulated character leaps higher into the air than the human when they perform jumps of about the same length. By moving the character's center of mass much farther forward of the area of support at take-off, a flatter trajectory could be produced. However, changing the trajectory would affect the landing controller, probably requiring that it be re-tuned to produce a successful landing.

8.2.1 Diving

I have created four 10 meter platform dives using the balance, leaping, and tumbling basis controllers. Two of the dives, the inward $1\frac{1}{2}$ somersault pike and backward $1\frac{1}{2}$ somersault with $\frac{1}{2}$ twist, are re-implementations of dives presented in [WH96]. The new implementation uses the basis controllers described earlier, instead of special purpose controllers for each dive. Both dives took about five hours of effort to create, whereas the special purpose controllers required about four weeks of work. The two dives for the male character can be seen in figure 8.5. Both dives used the same balance, leaping, and tumbling controllers but the inward dive leapt forward and performed a somersault while the twisting dive leapt backwards and used twists combined with somersaults.

Two new dives for the female character can be seen in figure 8.5. They are

Figure 8.4. Comparison of human broad jump and a simulated broad jump to a distance of 2.1 m. The time interval between frames is 0.33 s.

the inward $2\frac{1}{2}$ somersault pike and the backward $1\frac{1}{2}$ somersault with $2\frac{1}{2}$ twists. The inward dive required about five hours to create from an initial standing state, but the twisting dive took about fifteen hours because of the very high angular velocity about the twisting axis.

8.3 Back Handspring

The back handspring is a gymnastic maneuver that uses all four controllers (figure 8.6). The leaping controller initiates the maneuver by causing the character to jump backwards into the air with substantial rotation about the somersaulting axis. The tumbling controller performs a backwards $\frac{1}{2}$ somersault in the layout position. The character then lands on his hands and after a set amount of time pushes off the ground to perform another $\frac{1}{2}$ somersault. The landing controller positions the legs for touchdown and transitions to the balance controller to complete the maneuver.

The aspect of this maneuver that required the most tuning was generating sufficient backwards angular velocity to land in a handstand position. If the character

Figure 8.5. Filmstrips (read from top to bottom) demonstrating the inward $2\frac{1}{2}$ somersault pike and the backward $1\frac{1}{2}$ somersault with $2\frac{1}{2}$ twists for the female character and the inward $1\frac{1}{2}$ somersault pike and backward $1\frac{1}{2}$ somersault with $\frac{1}{2}$ twist for the male character. The time interval between frames is 0.66 s.

under-rotated, the center of mass would be positioned too far forward at landing; if he over-rotated the center of mass would be too far backwards. The hands were not controlled by the basis controllers, instead they were programmed to push off the ground a set amount of time after contact.

Transitions from one controller to the next are sometimes unsuccessful and a failure results, such as the one shown in figure 8.7. The character's feet were too far forward of the center of mass and he fell over backwards. A nice outcome of using simulation to generate motion is that the failures often provide useful insight into how the controllers and the transitions between them are failing. The failures also often look natural, in that a human making a similar mistake would fall down in a similar fashion. In this example, the male character appears to be reaching behind himself to catch himself in much the same way a human might. This action was not explicitly programmed but was just how the arms happened to move as the character fell over.

8.4 Standing Somersault

In this maneuver, the leaping controller generates the height and rotation rate required for either a forward or backward tucked somersault, the landing controller untucks and positions the legs for landing and then transitions to the balance controller. Figure 8.8 shows the male character performing a standing backward somersault. A standing somersault is more difficult to perform than a back handspring because the athlete does not use his or her hands, therefore the athlete must jump higher and with a greater rotation rate.

Using the four basis controllers, the initial version of a successful backward somersault required about three hours of tuning. The control parameters of the leaping controller were adjusted until the character attained a height and angular velocity

Figure 8.6. Filmstrip demonstrating the back handspring maneuver. The time interval between frames is 0.83 s.

Figure 8.7. Filmstrip demonstrating the male character failing to perform a successful back handspring. The time interval between frames is 0.83 s.

Figure 8.8. Filmstrip demonstrating the standing backward somersault for the male character. The time interval between frames is 0.66 s.

that made landing possible. Then the timing of the transition from the tumbling controller to the landing controller was modified until the maneuver was completed successfully. At first the maneuver did not look realistic because the character appeared to perform this difficult maneuver without significant effort. Lowering the desired leaping height and increasing the angular velocity made the behavior look more natural.

8.5 Forward Somersault

The first part of this section discusses the process of creating a new maneuver, the standing forward somersault, from a combined set of basis controllers. The second part analyzes the changes that must be made to the control parameters when the dynamic parameters of the model are altered by adding mass to a backpack on the female character.

8.5.1 Creating the Forward Somersault

The first step in creating a forward somersault was to have the character leap into the air and initiate a tucked, forward somersault (figure 8.9). Once the character attained a height and angular velocity that were reasonable for landing, the transition timing

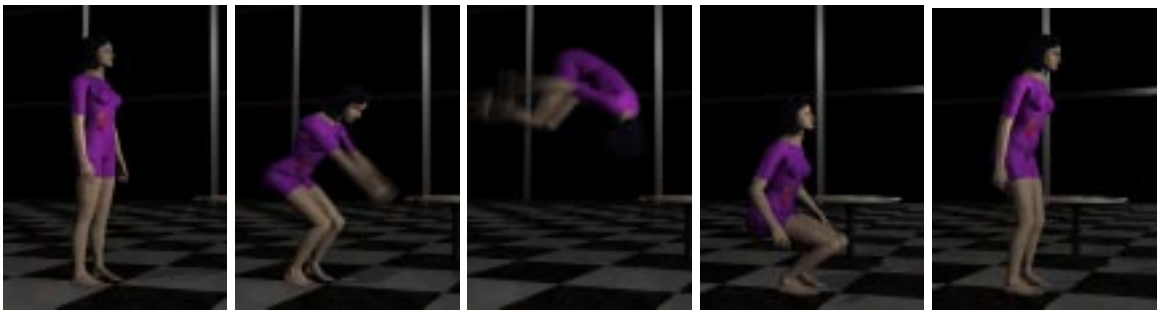


Figure 8.9. Filmstrip demonstrating the forward somersault for the female character. The time interval between frames is 0.66 s.

from the tumbling controller to the landing controller was tuned. In the forward somersault the hips and knees were extended slightly during the aerial maneuver before the landing controller became active.

The initial version of the forward somersault took about three hours to create. But the first attempt looked unnatural for two subtle reasons: the character's center of mass traveled backwards while the character somersaulted forward and the elbows and neck appeared stiff during landing. Making the character lean further forward before leaping into the air produced a forward velocity for the center of mass during the somersault and solved the first problem. Reducing the damping in the neck servos of the character and flexing the elbows before landing fixed the second. After landing, the elbows were slowly extended by the balance controller. Generating natural-looking motion required much more hand tuning than creating the initial behavior.

8.5.2 Changing Dynamic Parameters

One method of assessing the robustness of a controller is to change the dynamic parameters of the model and then record the changes that must be made to the controller in order to accomplish the given task. The addition of a 2.0 kg or 4.0 kg backpack to the female character was used to make a preliminary assessment of the robustness of the forward somersault.

With the added mass and no changes to the control parameters, the basis controllers for the forward somersault failed to perform the maneuver. To produce a forward somersault for the 2.0 kg model, the time interval for the transition from tumbling to landing was decreased by 0.07 s. To produce a forward somersault for the 4.0 kg model the time interval was decreased by 0.12 s and the knees were flexed 6 degrees before initiating the landing controller. It took two iterations to produce a successful landing for the 2.0 kg model and six iterations for the 4.0 kg model.

For qualitative comparison, data was collected from the successful completion of each forward somersault. The graphs in figure 8.10 compare the vertical component of the center of mass and the peak somersaulting angular velocity. The height of the center of mass increases because the backpack raises the character's center of mass. Increasing the mass caused a lower angular velocity because of the extra inertia from the increased mass of the backpack. The time difference in foot contact at landing between the 0.0 kg case and the 2.0 kg case was 0.015 s and between the 0.0 kg and 4.0 kg case was 0.02 s. The transition to balance in the recovery phase of the maneuver occured 0.43 s earlier in the 4.0 kg case than in 0.0 kg or 2.0 kg case (reflected by the height of the center of mass in figure 8.10) because the horizontal location of the center of mass was further forward.

From these tests we can conclude that as long as the parameters of the model do not change significantly, the basis controllers will require minimal changes. A search process like the one described by Hodgins and Pollard could be used to automatically tune the control system when the sizes and masses of links change significantly [HP97].

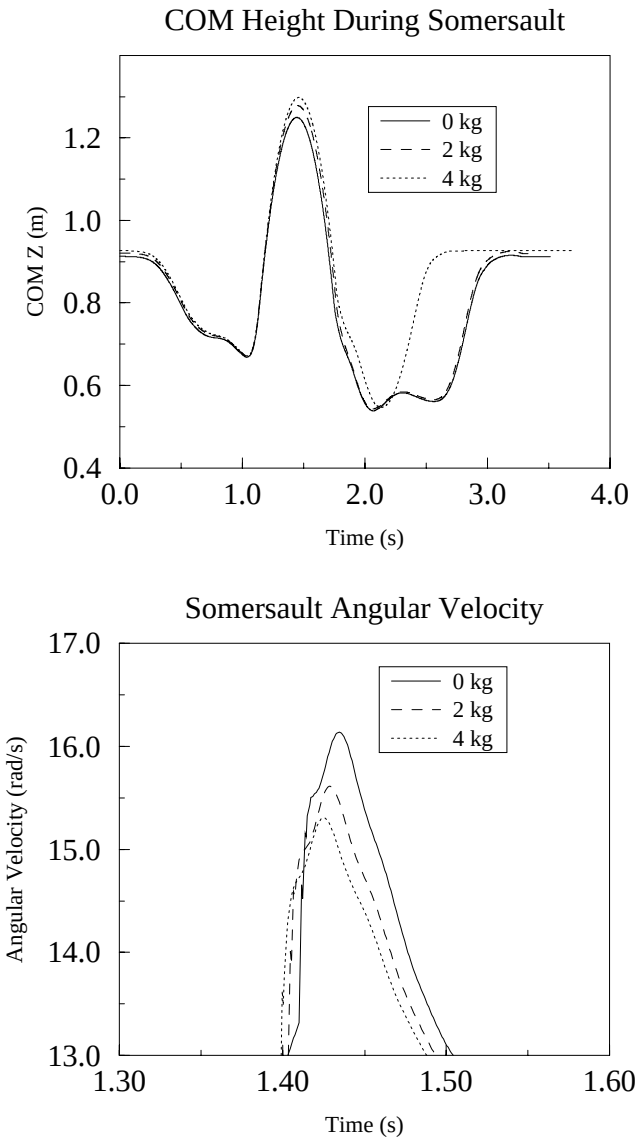


Figure 8.10. Graphs comparing the center of mass height and peak angular velocity of three characters performing a forward somersault. The characters had no extra mass, a 2.0 kg backpack, or a 4.0 kg backpack.

CHAPTER 9

Conclusion

The goal of this research was to develop basis controllers that allowed a variety of simulated behaviors to be created from a small set of simple, parameterized behaviors. I have developed control algorithms that allow an animator to generate a variety of dynamic behaviors by combining parameterized controllers for leaping, tumbling, landing, and balance. While these four controllers only represent a small subset of gymnastics, much less of all of human motion, a library of basis controllers with ten times as many behaviors would have sufficient functionality to be useful if the behaviors could be combined easily. With a sufficient number of basis controllers, new behaviors could be created by selecting the appropriate basis controllers and tuning their parameters to produce the desired behavior.

By parameterizing the controllers, I was able to construct more general and robust basis controllers. Instead of developing several leaping controllers for different styles of leaps, I built a single controller that produced horizontal and vertical leaps as well as leaps with significant angular momentum. Similarly, a single controller maintains balance independent of the upper body position selected by the user. I assessed the robustness of the controllers by measuring their performance for a range of input values and types of external disturbances. The generality of the behaviors is important because it provides a number of distinctly different paths through the state space and generates a wider variety of motion from a small set of behaviors. The robustness of the behaviors provides a measure of whether it will be possible to transition between a given pair of behaviors. For example, if the landing controller

can restore balance given an angular velocity of 12.56 rad/s at touchdown, then the tumbling controller can perform a 2π rotation during a 0.5 s aerial phase.

For the complex behaviors presented in this paper, I tuned the parameters of each controller by hand over a period of a few hours. However, these parameters may well be more amenable to automatic design than the full set of parameters for a complex dynamic behavior because the basis controllers contain substantial domain knowledge. For example, a search procedure should be able to easily find the exact body pitch at lift-off that will generate the required angular momentum for a forward flip. The success of the maneuver can be assessed automatically as part of an evaluation function by measuring how easily the landing controller can restore balance. Stylistic details such as arms that are too stiff will be harder to assess and tune without human input.

Because significantly more than four behaviors will be required to create a useful animation system, the question of how well this approach will scale is important. The funnels in state space that represent the four behaviors I developed fit together well in a particular order (balancing, leaping, tumbling, landing, and then balancing). Extending this approach to a more general set of basis behaviors will require adopting some of the features of the two other approaches (transitions between all pairs of behaviors and transitions between all behaviors and a home behavior as shown in the first two illustrations of figure 1.2). For dynamically simple behaviors such as gesturing, the approach of creating transitions between all pairs of behaviors is probably best because interpolation combined with collision avoidance should allow transitions to be generated automatically. For broad classes of dynamic behaviors, such as crawling and running, a modified version of the "home-position" approach probably makes the most sense. For example, a transition from running to crawling would require briefly performing intermediate behaviors such as walking, standing,

and crouching. Similarly, in the set of four behaviors, a transition from tumbling to balancing required the use of a landing controller to reduce the velocities of the aerial phase to within the capture region of the balance controller.

9.1 Future Work

A number of research problems must still be addressed before simulation of human motion becomes valuable for animators, video game players, or athletes. Open problems include developing a larger library of basis behaviors, making existing behaviors more robust and interactive, and building a more detailed human model.

Leaping, tumbling, landing, and balancing can be used to produce a number of gymnastic maneuvers, but many more behaviors will be needed to create a synthetic human that interacts with human participants in a three-dimensional virtual environment. Some of the controllers that would be required by synthetic agents include locomotion (both running and walking), sitting and rising, climbing stairs, and manipulation of objects in the environment. I envision that someday a first-person shooter style video game could be constructed that uses dynamic simulation instead of motion capture by incorporating these basis behaviors. I predict that using simulated behaviors for the video game would yield a much richer set of behaviors than the current motion capture games provide.

The number and granularity of basis behaviors that are required to simulate a wide variety of human motions is also an important issue. How many behaviors are needed to simulate the activities of a synthetic actor commuting to work in the morning? The answer depends on the granularity of the basis controllers. The controllers could be developed to accomplish broad tasks like driving in traffic, or they could be broken down into minute tasks, like depressing the accelerator, turning the steering wheel, and activating a turn signal. The best approach will probably depend

on the application. For example, interactive agents might be controlled with broad tasks and embedded high-level planners but animation applications might work better with narrow tasks. Animation applications might require adjustment of the finer details of motion, whereas interactive applications might find the overall goal of the behavior more important. I have concentrated on the production of animation in this thesis and the basis controllers have a fine degree of granularity.

To interact in a natural fashion with unpredictable human users, synthetic actors must use behaviors that perform well under a wide variety of conditions and disturbances. For example, the combination of controllers that produce a standing forward somersault should continue to produce a somersault if mass is added to the character. The landing controller should be able to take a step if necessary to transition to a balanced state. However, the controllers should not allow the simulations to perform unnatural actions. For example a human should not be able to leap 5 m into the air.

To be truly interactive, the motion of synthetic actors in virtual environments must be computed in real-time (simulation time must be less than wall clock time). The models presented in this research run 70 times slower than real-time on a Silicon Graphics Octane workstation with an R10000 processor. However, human models with fewer degrees of freedom (like the bicyclist developed by Hodgins [HWBO95]) already run in real-time on the same hardware. I anticipate that with improved dynamic simulation techniques and continued increase in workstation speed, a high degree-of-freedom, three-dimensional human simulation will run in real time within a few years.

A more accurate human model is another area for future research. The muscle model used in this research was very simple: a torque source at each joint. The strength of individual joints was not taken into account, and this simplification could

have produced a simulated human that was stronger or faster than a real human. For example, when the simulated character performed a broad jump it leapt much higher than the human subject. The absence of a strength model means the simulated characters could perform maneuvers using strategies that are impossible for humans. For example, the simulated character could produce more thrust from the ankles than a human in order to leap higher than the best human athletes ever could.

Simulated motion of athletic endeavors has the potential to be useful in improving athletic performance if an accurate and validated biomechanical model can be constructed. Interactive simulations of a human model that closely approximate a specific athlete's body could give both coaches and athletes better intuition about the physics involved in their sport and could lead to improved human performance. In this thesis I chose to model the whole human body without a high degree of biomechanical accuracy, but with sufficient accuracy so that the characters could generally perform the desired tasks in a natural-looking fashion. If this technique were to be used for predicting human performance, it would require a much more accurate model.

9.2 Conclusion

This thesis presents the results of research to find an improved method for controlling dynamically simulated human figures. Parameterized basis controllers were developed so that each controller could produce a variety of behaviors. Complex behaviors could be produced by combining basis controllers. As a demonstration of the power of this approach, four controllers were developed that allowed a simulated human character to leap, tumble, land, and balance. By combining the four controllers, animations of numerous diving and gymnastic maneuvers were created. Each of the basis controllers were derived from observations of human performance in an attempt to produce natural looking motion. Experiments were performed to assess the performance of

the individual controllers and combinations of controllers. The methods presented in this thesis bring us one step closer to creating a synthetic character that interacts with people in virtual environments, acts in a computer generated movie, or helps train an athlete.

APPENDIX A

Appendix

A.1 SD-Fast Input File for Male Model

Following is the SD-Fast input file used to derive the equations of motion for the 53 degree-of-freedom male model.

```
# SD/FAST System Description File
# for creature man

gravity = 0.0 0.0 -9.80665000

# printing free body : PELVIS
# original center of mass = -0.006808, 0.000377, 1.001726
body = PELVIS
  mass = 10.73192768
  inertia = 0.08625900 0.07114769 0.10951932

#printing tree-joint body : CHEST3
# with joint SPINEL3
# original center of mass = 0.015173, -0.000165, 1.128768
body = CHEST3  inboard = PELVIS  joint = gimbal
  mass = 6.16425603
  inertia = 0.04426617 0.02708728 0.06084866
  bodyToJoint = -0.01437270 0.00016502 -0.04986785
  inbToJoint  = 0.00760776 -0.00037653 0.07717409
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : ULEGR
# with joint HIPR
# original center of mass = -0.008070, -0.095635, 0.763461
body = ULEGR  inboard = PELVIS  joint = gimbal
  mass = 10.46012762
```

```
  inertia = 0.15355185 0.16367185 0.04429564
  bodyToJoint = -0.01112982 0.01363505 0.17363924
  inbToJoint  = -0.01239224 -0.08237653 -0.06462591
  pin = 0.0 1.0 0.0
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : ULEGL
# with joint HIPL
# original center of mass = -0.008117, 0.095659, 0.763682
body = ULEGL  inboard = PELVIS  joint = gimbal
  mass = 10.47768003
  inertia = 0.15406018 0.16421820 0.04440836
  bodyToJoint = -0.01108264 -0.01365917 0.17341847
  inbToJoint  = -0.01239224 0.08162347 -0.06462591
  pin = 0.0 1.0 0.0
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : LLEGR
# with joint KNEER
# original center of mass = -0.041461, -0.095533, 0.333228
body = LLEGR  inboard = ULEGR  joint = pin
  mass = 5.32342559
  inertia = 0.07455836 0.07841995 0.01133475
  bodyToJoint = 0.02146139 -0.00046737 0.19117168
  inbToJoint  = -0.01192982 -0.00036495 -0.23906076
  pin = 0.0 1.0 0.0

#printing tree-joint body : LLEGL
# with joint KNEEL
# original center of mass = -0.041456, 0.095573, 0.333268
body = LLEGL  inboard = ULEGL  joint = pin
  mass = 5.32382113
  inertia = 0.07456326 0.07842407 0.01133626
  bodyToJoint = 0.02145587 0.00042748 0.19113195
  inbToJoint  = -0.01188264 0.00034083 -0.23928153
  pin = 0.0 1.0 0.0

#printing tree-joint body : FOOTR
# with joint ANKLER
# original center of mass = -0.020427, -0.097594, 0.039208
body = FOOTR  inboard = LLEGR  joint = ujoint
  mass = 0.82704597
```

```
  inertia = 0.00079011 0.00169920 0.00170150
  bodyToJoint = -0.03407298 0.00929370 0.04439173
  inbToJoint  = -0.01303861 0.00723263 -0.24962832
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : FOOTL
# with joint ANKLEL
# original center of mass = -0.020427, 0.097646, 0.039199
body = FOOTL  inboard = LLEGL  joint = ujoint
  mass = 0.82708382
  inertia = 0.00079017 0.00169904 0.00170133
  bodyToJoint = -0.03407336 -0.00934602 0.04440053
  inbToJoint  = -0.01304413 -0.00727252 -0.24966805
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : TOER
# with joint MTR
# original center of mass = 0.084839, -0.102868, 0.018455
body = TOER  inboard = FOOTR  joint = pin
  mass = 0.27931170
  inertia = 0.00019601 0.00021530 0.00032521
  bodyToJoint = -0.03513905 -0.00723206 0.00374546
  inbToJoint  = 0.07012702 -0.01250630 -0.01700827
  pin = 0.0 1.0 0.0

#printing tree-joint body : TOEL
# with joint MTL
# original center of mass = 0.084836, 0.102927, 0.018452
body = TOEL  inboard = FOOTL  joint = pin
  mass = 0.27936107
  inertia = 0.00019611 0.00021532 0.00032534
  bodyToJoint = -0.03513646 0.00717282 0.00374795
  inbToJoint  = 0.07012664 0.01245398 -0.01699947
  pin = 0.0 1.0 0.0

#printing tree-joint body : CHEST2
# with joint SPINEL1
# original center of mass = 0.010476, 0.000097, 1.245442
body = CHEST2  inboard = CHEST3  joint = gimbal
  mass = 8.38627372
  inertia = 0.07367761 0.04328265 0.09559950
  bodyToJoint = -0.01167600 -0.00009735 -0.06524239
```

```
  inbToJoint  = -0.01637270 0.00016502 0.05143215
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : CHEST1
# with joint SPINET7
# original center of mass = -0.000434, 0.000066, 1.353378
body = CHEST1  inboard = CHEST2  joint = gimbal
  mass = 8.77688230
  inertia = 0.10357844 0.04943749 0.14010687
  bodyToJoint = -0.00156584 -0.00006612 -0.04937830
  inbToJoint  = -0.01247600 -0.00009735 0.05855761
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : STERNUM
# with joint SPINET4
# original center of mass = -0.011293, -0.000200, 1.468901
body = STERNUM  inboard = CHEST1  joint = gimbal
  mass = 2.19711680
  inertia = 0.00510826 0.01233765 0.00897859
  bodyToJoint = -0.00110665 0.00020018 -0.06970092
  inbToJoint  = -0.01196584 -0.00006612 0.04582170
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : CHESTR
# with joint CLAVR
# original center of mass = -0.009119, -0.104250, 1.464177
body = CHESTR  inboard = STERNUM  joint = ujoint
  mass = 4.08793607
  inertia = 0.01437074 0.02037179 0.02048838
  bodyToJoint = 0.01161879 0.06885024 0.01912323
  inbToJoint  = 0.01379335 -0.03519982 0.01439908
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : CHESTL
# with joint CLAVL
# original center of mass = -0.009121, 0.104279, 1.464204
body = CHESTL  inboard = STERNUM  joint = ujoint
```

```
  mass = 4.08715518
  inertia = 0.01440587 0.02039169 0.02046155
  bodyToJoint = 0.01162093 -0.06887917 0.01909569
  inbToJoint  = 0.01379335 0.03560018 0.01439908
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : UARMR
# with joint SHLDR
# original center of mass = 0.005265, -0.295709, 1.466786
body = UARMR  inboard = CHESTR  joint = ball
  mass = 3.82365595
  inertia = 0.02567638 0.00921546 0.02716907
  bodyToJoint = -0.00576475 0.11420892 0.01641385
  inbToJoint  = 0.00861879 -0.07724976 0.01902323

#printing tree-joint body : UARML
# with joint SHLDL
# original center of mass = 0.005250, 0.295693, 1.466802
body = UARML  inboard = CHESTL  joint = ball
  mass = 3.82079247
  inertia = 0.02565624 0.00919585 0.02714534
  bodyToJoint = -0.00574953 -0.11419286 0.01639848
  inbToJoint  = 0.00862093 0.07722083 0.01899569

#printing tree-joint body : LARMR
# with joint ELBR
# original center of mass = 0.057350, -0.590853, 1.478376
body = LARMR  inboard = UARMR  joint = ujoint
  mass = 1.77960435
  inertia = 0.01303935 0.00294703 0.01485761
  bodyToJoint = -0.04084986 0.13475268 -0.00377603
  inbToJoint  = 0.01123525 -0.16039108 0.00781385
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : LARML
# with joint ELBL
# original center of mass = 0.057842, 0.592476, 1.478385
body = LARML  inboard = UARML  joint = ujoint
  mass = 1.75104662
  inertia = 0.01253812 0.00287798 0.01431001
  bodyToJoint = -0.04134162 -0.13637649 -0.00378509
  inbToJoint  = 0.01125047 0.16040714 0.00779848
```

```
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : HANDR
# with joint WRSTR
# original center of mass = 0.143880, -0.831780, 1.457334
body = HANDR  inboard = LARMR  joint = ujoint
  mass = 0.66527452
  inertia = 0.00118359 0.00067766 0.00128598
  bodyToJoint = -0.02477955 0.07408025 0.01696596
  inbToJoint  = 0.06175014 -0.16684732 -0.00407603
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : HANDL
# with joint WRSTL
# original center of mass = 0.143888, 0.831750, 1.457274
body = HANDL  inboard = LARML  joint = ujoint
  mass = 0.66455597
  inertia = 0.00120281 0.00069644 0.00128522
  bodyToJoint = -0.02478783 -0.07404980 0.01702640
  inbToJoint  = 0.06125838 0.16522351 -0.00408509
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : HEAD
# with joint NECKC2
# original center of mass = 0.044730, -0.000033, 1.693290
body = HEAD  inboard = STERNUM  joint = gimbal
  mass = 7.10606697
  inertia = 0.04919430 0.05971314 0.02982596
  bodyToJoint = -0.03142963 0.00003337 -0.13849000
  inbToJoint  = 0.02459335 0.00020018 0.08589908
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#loop joint section
constraints = 12
```

A.2 SD-Fast Input File for Female Model

Following is the SD-Fast input file used to derive the equations of motion for the 53 degree-of-freedom female model.

```
# SD/FAST System Description File
# for creature woman

gravity = 0.0 0.0 -9.80665

# printing free body : PELVIS
# original center of mass = -0.027413, 0.000268, 0.887866
body = PELVIS
  mass = 6.32919497
  inertia = 0.03868504 0.02831247 0.04393691

#printing tree-joint body : CHEST3
# with joint SPINEL3
# original center of mass = -0.010423, 0.000105, 0.991410
body = CHEST3  inboard = PELVIS  joint = gimbal
  mass = 2.10200160
  inertia = 0.00834432 0.00433670  0.01115681
  bodyToJoint = -0.00557687 -0.00010479 -0.02941049
  inbToJoint  = 0.01141261 -0.00026837 0.07413422
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : ULEGR
# with joint HIPR
# original center of mass = -0.026838, -0.081858, 0.662290
body = ULEGR  inboard = PELVIS  joint = gimbal
  mass = 7.41113444
  inertia = 0.11154523 0.11546858 0.02203643
  bodyToJoint = -0.01116249 0.00485824 0.17370996
  inbToJoint  = -0.01058739 -0.07726837 -0.05186578
  pin = 0.0 1.0 0.0
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : ULEGL
# with joint HIPL
# original center of mass = -0.026832, 0.081929, 0.661700
```

```
body = ULEGL  inboard = PELVIS  joint = gimbal
  mass = 7.38199049
  inertia = 0.11072447 0.11463433 0.02190973
  bodyToJoint = -0.01116803 -0.00492935 0.17430041
  inbToJoint  = -0.01058739 0.07673163 -0.05186578
  pin = 0.0 1.0 0.0
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : LLEGR
# with joint KNEER
# original center of mass = -0.038985, -0.087549, 0.262613
body = LLEGR  inboard = ULEGR  joint = pin
  mass = 1.99922397
  inertia = 0.01500744 0.01518789 0.00206030
  bodyToJoint = 0.01198529 0.00954947 0.13838679
  inbToJoint  = -0.00016249 0.00385824 -0.26129004
  pin = 0.0 1.0 0.0

#printing tree-joint body : LLEGL
# with joint KNEEL
# original center of mass = -0.038993, 0.087538, 0.262614
body = LLEGL  inboard = ULEGL  joint = pin
  mass = 1.99896813
  inertia = 0.01500421 0.01518365 0.00205986
  bodyToJoint = 0.01199311 -0.00953830 0.13838563
  inbToJoint  = -0.00016803 -0.00392935 -0.26069959
  pin = 0.0 1.0 0.0

#printing tree-joint body : FOOTR
# with joint ANKLER
# original center of mass = -0.014792, -0.093798, 0.036262
body = FOOTR  inboard = LLEGR  joint = ujoint
  mass = 0.51208883
  inertia = 0.00039442 0.00073929 0.00065925
  bodyToJoint = -0.02420792 0.00579759 0.04573842
  inbToJoint  = -0.00001471 -0.00045053 -0.18061321
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : FOOTL
# with joint ANKLEL
# original center of mass = -0.014794, 0.093800, 0.036262
body = FOOTL  inboard = LLEGL  joint = ujoint
```

```
  mass = 0.51200006
  inertia = 0.00039428 0.00073911 0.00065909
  bodyToJoint = -0.02420616 -0.00580036 0.04573784
  inbToJoint  = -0.00000689 0.00046170 -0.18061437
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : TOER
# with joint MTR
# original center of mass = 0.072825, -0.086748, 0.013514
body = TOER  inboard = FOOTR  joint = pin
  mass = 0.15297542
  inertia = 0.00007656 0.00007881 0.00012469
  bodyToJoint = -0.02982516 -0.00725216 0.00148575
  inbToJoint  = 0.05779208 -0.00020241 -0.02126158
  pin = 0.0 1.0 0.0

#printing tree-joint body : TOEL
# with joint MTL
# original center of mass = 0.072825, 0.086755, 0.013509
body = TOEL  inboard = FOOTL  joint = pin
  mass = 0.15297222
  inertia = 0.00007660 0.00007881 0.00012474
  bodyToJoint = -0.02982516 0.00724545 0.00149124
  inbToJoint  = 0.05779384 0.00019964 -0.02126216
  pin = 0.0 1.0 0.0

#printing tree-joint body : CHEST2
# with joint SPINEL1
# original center of mass = -0.013545, 0.000112, 1.071086
body = CHEST2  inboard = CHEST3  joint = gimbal
  mass = 2.62236597
  inertia = 0.01026760 0.00640238 0.01355024
  bodyToJoint = -0.00845506 -0.00011161 -0.04408629
  inbToJoint  = -0.01157687 -0.00010479 0.03558951
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : CHEST1
# with joint SPINET7
# original center of mass = -0.022843, 0.000342, 1.158771
body = CHEST1  inboard = CHEST2  joint = gimbal
  mass = 4.27710072
```

```
  inertia = 0.02356694 0.00003799 0.00026791
            ?          0.01582748 -0.00001199
            ?          ?          0.03369183
  bodyToJoint = -0.00815706 -0.00034165 -0.04777056
  inbToJoint  = -0.01745506 -0.00011161 0.03991371
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : STERNUM
# with joint SPINET4
# original center of mass = -0.046480, -0.000030, 1.270140
body = STERNUM  inboard = CHEST1  joint = gimbal
  mass = 1.20288477
  inertia = 0.00255317 0.00544391 0.00331153
  bodyToJoint = 0.01147995 0.00002965 -0.06913993
  inbToJoint  = -0.01215706 -0.00034165 0.04222944
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : CHESTR
# with joint CLAVR
# original center of mass = -0.046724, -0.086117, 1.264862
body = CHESTR  inboard = STERNUM  joint = ujoint
  mass = 3.07161686
  inertia = 0.00974820 0.01216337 0.01205795
  bodyToJoint = 0.01172404 0.06311748 0.03513842
  inbToJoint  = 0.01147995 -0.02297035 0.02986007
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : CHESTL
# with joint CLAVL
# original center of mass = -0.046841, 0.086225, 1.265165
body = CHESTL  inboard = STERNUM  joint = ujoint
  mass = 3.07182394
  inertia = 0.00975929 0.01213742 0.01213174
  bodyToJoint = 0.01184086 -0.06322476 0.03483491
  inbToJoint  = 0.01147995 0.02302965 0.02986007
  pin = 1.0 0.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : UARMR
```

```
# with joint SHLDR
# original center of mass = -0.051562, -0.295727, 1.265251
body = UARMR  inboard = CHESTR  joint = ball
  mass = 1.82684296
  inertia = 0.01220178 0.00219215 0.01243861
  bodyToJoint = -0.01443781 0.12472698 0.02174872
  inbToJoint  = -0.01927596 -0.08488252 0.02213842

#printing tree-joint body : UARML
# with joint SHLDL
# original center of mass = -0.051905, 0.294703, 1.264776
body = UARML  inboard = CHESTL  joint = ball
  mass = 1.85585454
  inertia = 0.01239712 0.00225052 0.01262077
  bodyToJoint = -0.01409531 -0.12370283 0.02222437
  inbToJoint  = -0.01915914 0.08477524 0.02183491

#printing tree-joint body : LARMR
# with joint ELBR
# original center of mass = -0.005165, -0.538935, 1.235356
body = LARMR  inboard = UARMR  joint = ujoint
  mass = 0.74929802
  inertia = 0.00270968 0.00074898 0.00301057
  bodyToJoint = -0.03083470 0.08893459 0.00864379
  inbToJoint  = 0.01556219 -0.15427302 -0.02125128
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : LARML
# with joint ELBL
# original center of mass = -0.003714, 0.541634, 1.234393
body = LARML  inboard = UARML  joint = ujoint
  mass = 0.78526077
  inertia = 0.00300027 0.00082843 0.00333567
  bodyToJoint = -0.03228567 -0.09163420 0.00960664
  inbToJoint  = 0.01590469 0.15529717 -0.02077563
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#printing tree-joint body : HANDR
# with joint WRSTR
# original center of mass = 0.054932, -0.717865, 1.193023
body = HANDR  inboard = LARMR  joint = ujoint
  mass = 0.33399683
```

```
  inertia = 0.00041227 0.00026027 0.00040049
  bodyToJoint = -0.01693224 0.05186523 0.02697691
  inbToJoint  = 0.04316530 -0.12706541 -0.01535621
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : HANDL
# with joint WRSTL
# original center of mass = 0.055743, 0.719930, 1.192218
body = HANDL  inboard = LARML  joint = ujoint
  mass = 0.31935170
  inertia = 0.00036701 0.00024317 0.00035605
  bodyToJoint = -0.01774276 -0.05393011 0.02778205
  inbToJoint  = 0.04171433 0.12436580 -0.01439336
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0

#printing tree-joint body : HEAD
# with joint NECKC2
# original center of mass = -0.046347, 0.000026, 1.486782
body = HEAD  inboard = STERNUM  joint = gimbal
  mass = 4.63528342
  inertia = 0.02167552 0.02625140 0.01435155
  bodyToJoint = -0.01565287 -0.00002632 -0.09308243
  inbToJoint  = -0.01552005 0.00002965 0.12356007
  pin = 1.0 0.0 0.0
  pin = 0.0 1.0 0.0
  pin = 0.0 0.0 1.0

#loop joint section
constraints = 12
```

Bibliography

[AGL87] W. W. Armstrong, M. Green, and R. Lake. Near-real-time control of human figure models. *IEEE Computer Graphics and Applications*, 7(6):52–61, June 1987.

[Bau72] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

[BC89] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 233–242, July 1989.

[BN88] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 309–315, August 1988.

[BPW93] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, 1993.

[BRK95] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Toward a systems theory for the composition of dynamically dexterous robot behaviors. *ISRR '95: Seventh International Symposium on Robotics Research*, 1995.

[CB95] B. L. Caster and B. T. Bates. The assesment of mechanical and neuromuscular response strategies during landing. *Medicine and Science in Sports and Exercise*, 27(5):736–744, 1995.

[Coh92] M. F. Cohen. Interactive spacetime control for animation. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 293–302, July 1992.

[Cou98] Singapore Sports Council. NAPFA scheme standards for males and females. In *http://www.ssc.gov.sg/sscphy3.htm*, January 1998.

[CS89] J. F. Cremer and J. A. Stewart. The architecture of newton, a gerneral-purpose dynamics simulator. In *IEEE International Conference on Robotics and Automation*, pages 1806–1811, May 1989.

[CS95] L. S. Crawford and S. S. Sastry. Biological motor control approaches for a planar diver. In *1995 Conference on Decision and Control*, December 1995.

[Dem55] W. T. Dempster. Space requirements of the seated operator. Technical Report WADC Technical Report 55-159, Wright Air Development Center, July 1955.

[DG65] W. T. Dempster and G. R. L. Gaughran. Properties of body segments based on size and weight. *American Journal of Anatomy*, 120:33–54, 1965.

[Fro79] C. Frohlich. Do springboard divers violate angular momentum conservation? *American Journal of Physics*, 47:583–592, 1979.

[GH23] H. S. Gasser and A. V. Hill. The dynamics of muscular contractions. *Royal Society of London Proceedings*, 96:398–437, 1923.

[Ghe77] C. Ghez. *Principles of Neurology*, chapter 39, pages 596–607. McGraw-Hill, New York, 1977.

[Gir87] M. Girard. Interactive design of 3D computer-animated legged animal motion. *IEEE Computer Graphics and Applications*, 7(6):39–51, June 1987.

[GT95] R. Grzeszczuk and D. Terzopoulos. Automated learning of Muscle-Actuated locomotion through control abstraction. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 63–70. ACM SIGGRAPH, August 1995.

[Hod91] J. K. Hodgins. Biped gait transitions. In *IEEE International Conference on Robotics and Automation*, pages 2092–2097, May 1991.

[Hod94] J. K. Hodgins. Simulation of human running. In *IEEE International Conference on Robotics and Automation*, pages 1320–1325, May 1994.

[HP97] J. K. Hodgins and N. S. Pollard. Adapting simulated behaviors for new characters. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 153–162. ACM SIGGRAPH, August 1997.

[HR90] J. K. Hodgins and M. H. Raibert. Biped gymnastics. *International Journal of Robotics Research*, 9(2):115–132, 1990.

[HR91] J. K. Hodgins and M. H. Raibert. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation*, 7(3):289–298, 1991.

[HRS91] M. Hollars, D. Rosenthal, and M. Sherman. *SD-Fast Users's Manual*. Symbolic Dynamics, Mountain View, 1991.

[HSL92] J. K. Hodgins, P. K. Sweeney, and D. G. Lawrence. Generating natural-looking motion for computer animation. In *Proceedings of Graphics Interface '92*, pages 265–272, May 1992.

[HWBO95] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, pages 71–78, August 1995.

[IC87] P. M. Isaacs and M. F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 215–224, July 1987.

[KB93] H. Ko and N. I. Badler. Straight line walking animation based on kinematic generalization that preserves the original characteristics. In *Proceedings of Graphics Interface '93*, pages 9–16, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[LGC94] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 35–42, July 1994.

[LK84] S. Lien and J. T. Kajiya. A symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra. *IEEE Computer Graphics and Applications*, 4(10):35–41, 1984.

[LvF96] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH '96*, pages 155–162, August 1996.

[LWZB90] P. Lee, S. Wei, J. Zhao, and N. I. Badler. Strength guided motion. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 253–262, August 1990.

[MG93] J. L. McNitt-Gray. Kinetics of the lower extremities during drop landings from three heights. *Journal of Biomechanics*, 26(9):1037–1046, 1993.

[MK93] N. Murthy and S. Keerthi. Optimal control of a somersaulting platform diver. In *IEEE International Conference on Robotics and Automation*, pages 1013–1018, May 1993.

[MZ90] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 29–38, August 1990.

[MZ96] M. McKenna and D. Zeltzer. Dynamic simulation of a complex human figure model with low level behavior control. *Presence*, 5(4):431–456, 1996.

[Nas76] L. M. Nashner. Adapting reflexes controlling the human posture. *Experimental Brain Research*, 26:59–72, 1976.

[Nas83] L. M. Nashner. Analysis of movement control in man using the movable platform. *Advances in Nuerology*, 39:607–619, 1983.

[NM93] J. T. Ngo and J. Marks. Spacetime constraints revisited. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 343–350, August 1993.

[OB88] H. N. Özgüven and N. Berme. An experimental and analytical study of impact forces during human jumping. *Journal of Biomechanics*, 21(12):1061–1066, 1988.

[Pai90] D. Pai. Programming anthropoid walking: Control and simulation. Technical Report 90-1178, Cornell Computer Science, 1990.

[PR92a] R. R. Playter and M. H. Raibert. Control of a biped somersault in 3-d. In *IEEE International Conference on Robotics and Automation*, pages 582–589, May 1992.

[PR92b] R. R. Playter and M. H. Raibert. Control of a biped somersault in 3-d. In *International Symposium on Theory of Machines and Mechanisms*, Sept 1992.

[PZSL90] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine. An optimal model for maximum-height human jumping. *Journal of Biomechanics*, 23(12):1185–1198, 1990.

[Rai86] M. H. Raibert. *Legged Robots That Balance*. MIT Press, Cambridge, 1986.

[RCJ82] M. H. Raibert, M. Chepponis, and H. B. Brown Jr. Running on four legs as though they were one. *IEEE Journal of Robotics and Automation*, 2(2):70–82, 1982.

[RH91] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 349–358, July 1991.

[RS86] D. E. Rosenthal and M. A. Sherman. High performance multibody simulations via symbolic equation manipulation and kane's method. *Journal of Astronautical Sciences*, 34(3):223–239, 1986.

[SC92] A. J. Stewart and J. F. Cremer. Beyond keyframing: An algorithmic approach to animation. In *Proceedings of Graphics Interface '92*, pages 273–281, May 1992.

[Sim94a] K. Sims. Evolving 3d morphology and behavior by competition. In *Artificial Life IV*, pages 28–39, 1994.

[Sim94b] K. Sims. Evolving virtual creatures. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, pages 15–22. ACM SIGGRAPH, July 1994.

[TT94] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, pages 43–50. ACM SIGGRAPH, July 1994.

[van96] M. van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, 16(2):40–49, March 1996.

[vF93] M. van de Panne and E. Fiume. Sensor-actuator networks. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 335–342, August 1993.

[vFV90] M. van de Panne, E. Fiume, and Z. Vranesic. Reusable motion synthesis using state-space controllers. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 225–234, August 1990.

[vFV92] M. van de Panne, E. Fiume, and Z. G. Vranesic. A controller for the dyamic walk of a biped across variable terrain. In *Proceedings of the 31st Conference on Decision and Control*, pages 2668–2673, December 1992.

[vSBv93] A. J. van Soest, A. L. Schwab, M. F. Bobbert, and G. J. van Ingen Schenau. The influence of the biarticularity of the gastrocnemius muscle on vertical-jumping achievment. *Journal of Biomechanics*, 26(1):1–8, 1993.

[WH95] W. L. Wooten and J. K. Hodgins. Simulation of human diving. In *Proceedings of Graphics Interface '95*, pages 1–9, Quebec City, Quebec, Canada, May 1995. Canadian Information Processing Society.

[WH96] W. L. Wooten and J. K. Hodgins. Animation of human diving. *Computer Graphics Forum*, 15(1):3–13, 1996.

[Wil87] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, June 1987.

[WK88] A. Witkin and M. Kass. Spacetime constraints. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 159–168, August 1988.

[WPF88] D. A. Winter, A. E. Patla, and J. S. Frank. Assessment of balance control in humans. *Journal of Biomechanics*, 21(12):1061–1066, 1988.

[YAH90] M. R. Yeadon, J. Athia, and F. D. Hales. The simulation of aerial movement-IV. *Journal of Biomechanics*, 23(1):85–89, 1990.

[Yea90] M. R. Yeadon. The simulation of aerial movement-I,II,III. *Journal of Biomechanics*, 23(1):59–83, 1990.